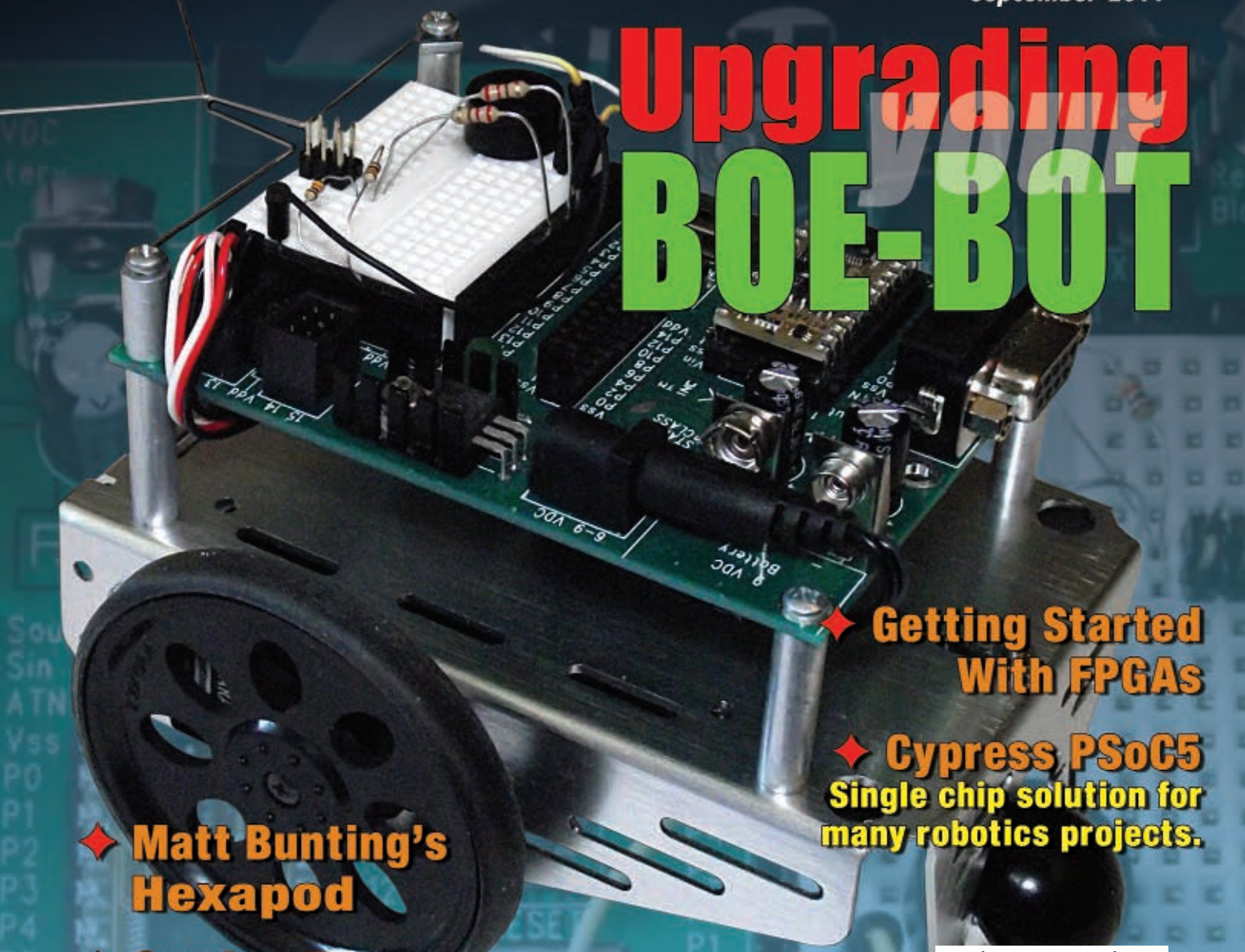
Design • Build • Program • Learn • Compete

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

September 2011



Upgrading BOE-BOT

- Getting Started With FPGAs
- Cypress PSoC5
  Single chip solution for many robotics projects.
- Matt Bunting's Hexapod
- Combat Zone
  - Fighting In Outdoor Arenas
  - Gear Terminology
  - NRL Championships

U.S. $5.50 CANADA $7.00

09

0 71486 02422 4

Let the Battle Begin!

Blast the competition with our giant **HS-5765MH** high voltage servo! Armed with **347 oz-in** of torque and heavy-duty, metal gears, this tough servo will make your bot the master of the combat arena! Its 2-cell LiPo capability and a 10mm output shaft provide the strength and speed required for the most demanding applications.

Get in the Ring!
Get Hitec!

6.0 Volts		7.4 Volts	
Speed	Torque	Speed	Torque
0.16	278 oz-in	0.13	347 oz-in

Part#	Dimensions	Weight
35765S	2.32 x 1.14 x 2.04 in	6.07 oz

HITEC

12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

do more wire less

Wixel

actual size

USB. Wireless. Programmable. All for $19.95.

The Wixel is a programmable microcontroller module with integrated USB and a 2.4 GHz radio. Write your own program or load pre-compiled, open-source apps to give your next project a wireless serial link, create a remote sensor network, and much more!

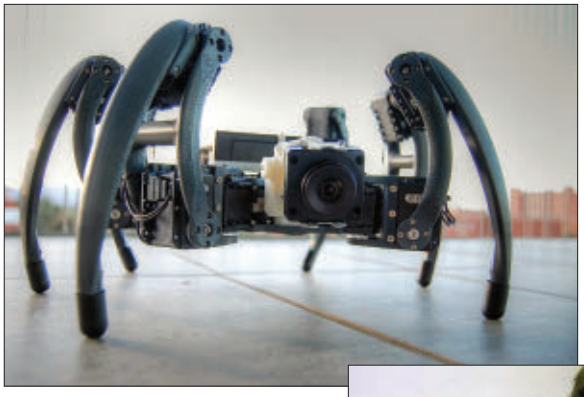
Pololu
Engage Your Brain

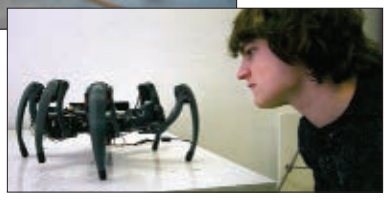
learn more at www.pololu.com/wixel

SERVO MAGAZINE

09.2011

VOL. 9 NO. 9





PAGE 10

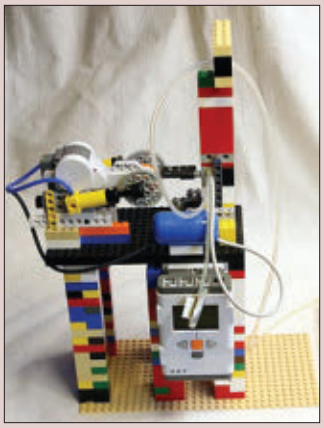
The Combat Zone...

Features

Events

Columns

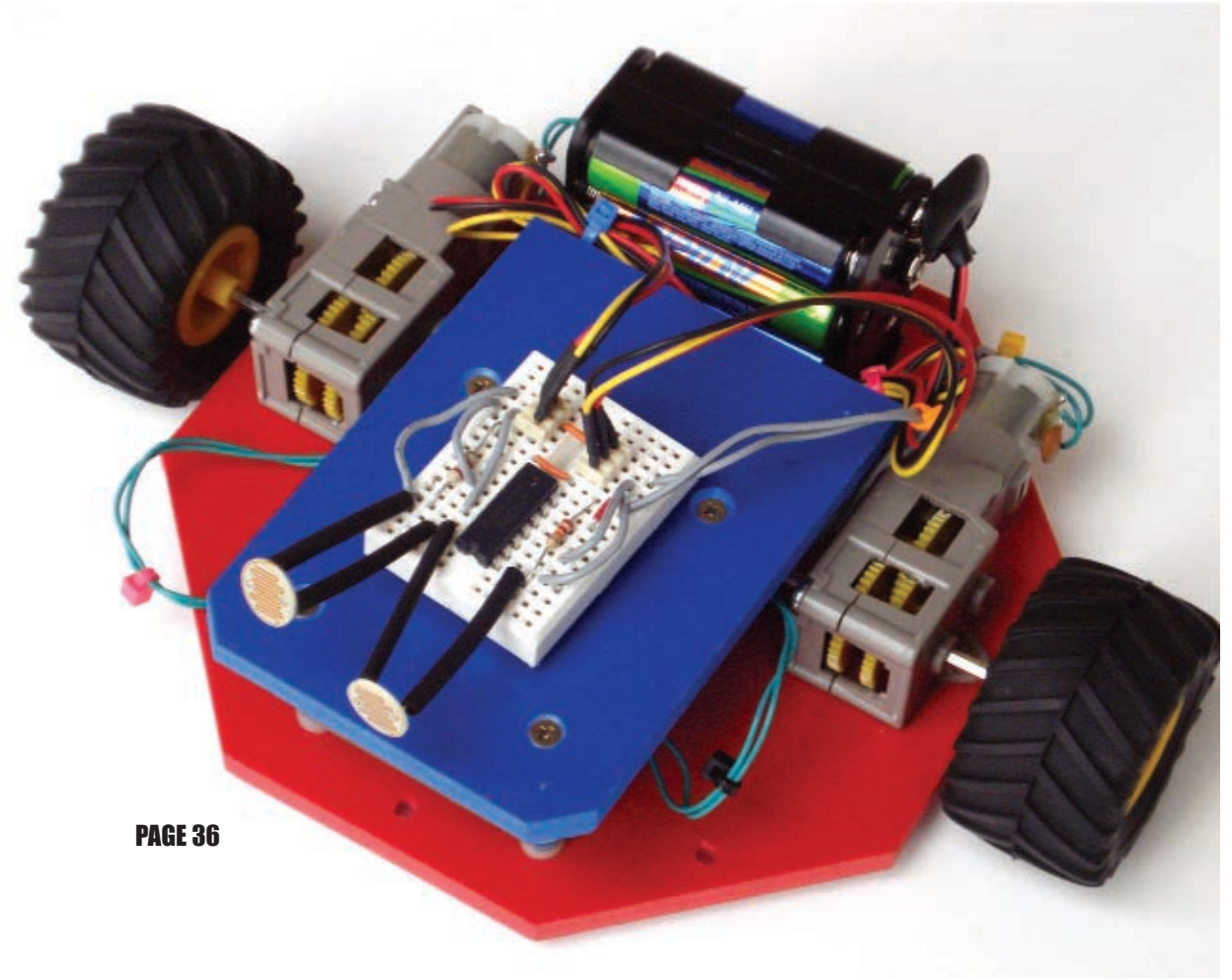
Departments



PAGE 22

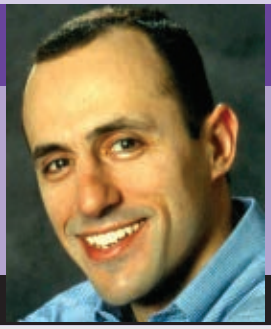
SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



PAGE 36

Mind / Iron



by Bryan Bergeron, Editor

Momentum

As roboticists, we're used to dealing with momentum — that is, the product of mass and velocity of a robot arm or other object. However, when it comes to success in design and construction, just as important is the momentum of your actions — the product of focus and activity that enables you to complete the projects you start. Too often, projects languish on a bookshelf or shop table, half completed. Despite your best intentions, some of your robotics projects may never see the light of day or reach their full potential, simply for lack of momentum.

I was reminded recently of this common tendency to lose momentum by an examiner at the US Patent and Trademark Office in Virginia. I was at the office to help the examiner decide which of two patents was valid. As you probably know, it's relatively easy for someone to break a patent if the inventor doesn't follow the rules of the game.

One of the rules is that the inventor has to show evidence of momentum. For example, let's say you've invented a new robotics drive mechanism based on magnetic repulsion. Once you've documented your invention, you can't simply apply for a patent and then put your notebook away, intending to revisit it some time in the future. To the contrary, if you want your patent to stick in the face of competition from another inventor, you have to prove to the patent office that you've been diligently perfecting your idea since you conceived of it. Human nature being what it is, the examiners in the US Patent and Trademark Office know that unless inventions are acted upon continuously, they will never result in practical products. It's in the national interest to award and validate patents that are likely to result in jobs and business.

So, whether you plan to make a million dollars with your new robot design or simply want to get the most out of your project hours, keep working toward your goals and stay focused. Resist the urge to put your current robotics project down until you've finished it. Of course, that's easier said than done.

I've found two successful approaches to keeping momentum going. The first assumes you have a fixed time for robotics every day or every weekend. Using an electronic spreadsheet or pencil and paper, block off one hour segments dedicated to particular tasks. For example, an hour for programming every weekday and three hours for robotics arm construction on the weekend.

Again, stay on the project and resist the temptation to work on something else. Sometimes this is difficult, especially when you reach a temporary impasse. However, when I'm stuck, I find it better to go for a walk or engage in some activity totally unrelated to robotics and then return with a fresh perspective.

The second approach to keeping momentum going is to make an ordered list of tasks that must be performed to finish your project. This approach is more useful when you can't plan your day in one hour blocks, but simply have to make use of the time when it's available. I like to use lists of stages of related development — say, mechanical platform assembly, which is broken down into drive train assembly servo mounting, and testing. Only when you've finish one area should you think about the next.

As with the time-based method,

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert
Tom Carroll, David Geer
Dennis Clark, R. Steven Rainwater
Gregory Intermaggio, Kevin Berry
Gordon McComb, David Ward
Lloyd Moore, William Henning
Nicholas Ozorak, James Baker
Morgan Berry, Bryce Woolley
Evan Woolley

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

Copyright 2011 by
T & L Publications, Inc.
All Rights Reserved

All advertising is subject to publisher's approval. We are not responsible for mistakes, misprints, or typographical errors. *SERVO Magazine* assumes no responsibility for the availability or condition of advertised items or for the honesty of the advertiser. The publisher makes no claims for the legality of any item advertised in *SERVO*. This is the sole responsibility of the advertiser. Advertisers and their agencies agree to indemnify and protect the publisher from any and all claims, action, or expense arising from advertising placed in *SERVO*. Please send all editorial correspondence, UPS, overnight mail, and artwork to: **430 Princeland Court, Corona, CA 92879.**

Printed in the USA on SFI & FSC stock.

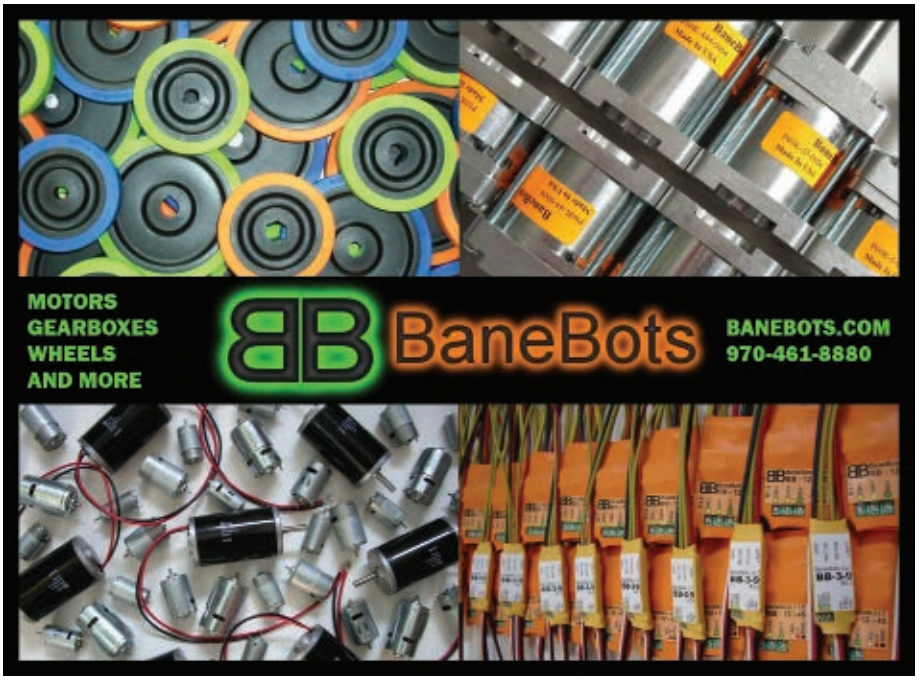
the name of the game is focus and definiteness of purpose. Resist the natural urge to bounce around from one type of task to the next. If you're programming, stay with it until you're done. Don't start a new subtask until you've finished the preceding tasks.

If you're working with a team – say a school robotics club – then you have the benefit of extra pairs of hands. However, without a concrete plan with agreed upon events and times, large development can dissolve into chaos. If you're working in a group, consider commercial planning software to help keep the momentum going. My favorite 'affordable' project management software is OmniPlan ($150), available as a download from **www.onmigroup.com**. Although OmniPlan is relatively inexpensive as project management software goes, the cost can be significant for a small robotics club. Whatever tool you use, just remember to stay focused and keep moving in the direction of your goals. SV

BIO-FEEDBACK

In the August edition of SERVO Magazine, we published "Maker Breaker," a rundown of some excellent robot-related events and exhibitions at the Maker Faire in San Mateo this May. I erroneously stated that the Western Region Robotics Forum (WRRF) booth was sponsored by FIRST (For Inspiration and Recognition of Science and Technology). The game exhibited was, in fact, "Logomotion" which was designed by FIRST. The booth itself, however, was being managed by WRRF, who is independent from FIRST.

Greg Intermaggio, writer




Robytes

by Jeff and Jenn Eckert

Are Self-Driven Cars Imminent?

Several companies have been backing the development of autonomous cars, including Volkswagen, Toyota, Ford, and GM, plus several universities and (somewhat curiously) Google. It may be a long time before such self-driving vehicles are common on US highways — but maybe not. Apparently, a Google-developed automated Toyota Prius has already logged more than 140,000 miles, traveling on the Pacific Coast highway, around Lake Tahoe, and even down Lombard St. In fact, the state of Nevada recently passed Bill 511 (**www.leg.state.nv.us/Session/76th2011/Bills/AB/AB511_EN.pdf**), "AN ACT relating to transportation; providing certain privileges to the owner or long-term lessee of a qualified alternative fuel vehicle; authorizing in this State the operation of, and a driver's license endorsement for operators of, autonomous vehicles; providing a penalty; and providing other matters properly relating thereto." Now we just have to figure out where to buy one.

GM's Electric Networked-Vehicle (EN-V) self-driving concept car.

Marathon Target robots like this one are designed to "make your day."

Pot Shot Bot

A pretty cool robotic application developed by the Aussies at Marathon Robotics (**www.marathon-targets.com**) takes the form of the Marathon Target machines. Not only are they highly valuable for marksmanship training programs, but someone has finally found a useful application for Segway® vehicles, which are employed for locomotion. According to the company, "Traditional targetry systems such as rails and pop-ups are fundamentally limited and inflexible in presenting realistic challenging targets. Marathon Targets has created smart targets — an innovative solution which eliminates the current constraints and introduces a step change in quality of training." The bots can move freely and randomly around the entire training environment, executing complex scenarios as determined by the on-board software, thus producing human-like movement. They use GPS technology and scanning laser rangefinders for navigation, positioning, and obstacle detection and avoidance. When one is shot, its mannequin drops, and the rest scatter and run for cover. The entire package is heavily armored and fitted with puncture-proof tires, and its motors can propel it at human running speed. The most obvious uses are for law enforcement and military training, and, in fact, the US Marines bought $50 million worth of them last year. But you have to think — wouldn't it be fun to put one in your backyard, invite some gun-toting redneck friends over, and crack a bottle of Jack Daniel's? Lock 'n load, Bubba! To see what it's all about, point your browser at **www.jkeckert.com/freedownloads/marathon.wmv**.

"Yotel" Features Robotic Bellhop

If you're headed for the Big Apple and want to stay in a place that is both (relatively) cheap and radically different, consider the Yotel (**www.yotel.com**), located just two blocks from Times Square. Whereas the average hotel room in NYC goes for $236 a night, Yotel rooms start at $179 and provide some luxurious amenities, even if in a small package. Contributing to the low overhead is the fact that check-in and check-out are automated via airport-like kiosks, and a 20 ft industrial robot handles your luggage. Plus, the basic room (referred to as a "cabin") is furnished with a futon-like bed and gives you only about 170 sq ft of breathing room. (For a few extra bucks, though, you can upgrade to a 320 sq ft unit with a king bed.) On the positive side, all cabins include "super strength" Wi-Fi and a workstation; a "technowall with TV, music, and power; a "monsoon power" shower room with body wash; and free breakfast on each floor. The facility features New York's largest outdoor terrace (7,000 sq ft), an indoor "studiyo," four bars, and a restaurant inspired by a sumo wrestling ring. This is the first US-based facility, but others are open for business at airports in London and Amsterdam.

The Yobot luggage robot handles the front desk at New York's Yotel.

The Bioloid robot with 31 hexagonal sensor modules attached to its body. Photo credit: Heddergott/TUM.

Sensitive Skin for Bots? Well ...

When the news piece came in with a headline that read, "Scientists Develop Sensitive Skin for Robots," it sounded like a long-anticipated breakthrough. Sadly, the reality is not quite so exciting, as the "skin" doesn't yet resemble skin, and all we're talking about is a 5 cm^2 circuit board with a bunch of sensors on it. But beyond the hype, it's still a noteworthy step in the right direction. Researchers at the Excellence Cluster Cognition for Technical Systems (CoTeSys) department (**www.cotesys.de**) of the Technical University of Munich (TUM) have assembled an artificial nerve package that eventually should enable robots "to feel heat or gentle touching on their surfaces." The sensors convey information about temperature, pressure, shear force, and vibration to augment what can be deduced via a bot's standard array of cameras.

Each board sports 15 sensors, including four IR units that detect anything closer than 1 cm, six temperature sensors, and an accelerometer. According to TUM engineer Philip Mittendorfer, "We try to pack many different sensory modalities into the smallest of spaces. In addition, it is easy to expand the circuit boards to later include other sensors; for example, pressure."

The information is fed to the central processor that enables each module to serve as a data hub for different sensory elements. Eventually, many circuits will be joined in a honeycomb planar structure that will cover appropriate parts of the bot. "We will close the skin and generate a prototype which is completely enclosed with these sensors, and can interact anew with its environment," according to TUM. "Beyond this, these machines will some day be able to incorporate our fundamental neurobiological capabilities and form a self-impression. The robot has moved a step closer to humanity." For a short vid, visit **www.youtube.com/watch?v=5CILOcxjkQY**.

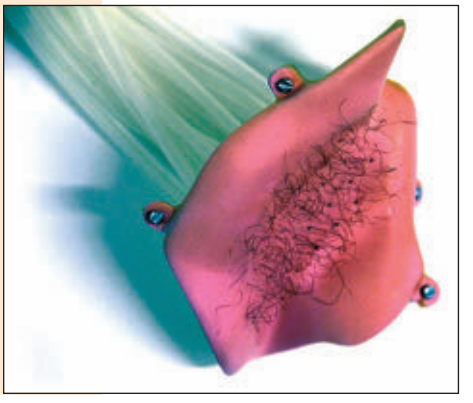
Art Meets Bots Again

Perhaps the latest example of art imitating machinery imitating life is the prototype robot armpit — a creation of London-based designer Kevin Grennan (**www.kevingrennan.com**). Once again illustrating the dangers of mixing fine arts with robotics, Grennan (who is studying for an MA in Design Interactions at the Royal College of Art) expounded, "What should a robot smell like? I have augmented three existing industrial robots with 'sweat glands.' Each uses a specific property of human subconscious behavior in response to a chemical stimulus. The contrast between the physical anti-anthropomorphic nature of the machines and the olfactory anthropomorphism highlights the absurd nature of the trickery at play in all anthropomorphism."

Thank goodness someone finally cleared that up. The website displays diagrams of the three bots, but there is no evidence that they actually exist outside of the creator's mind. But I'm not complaining; I'm just glad he didn't decide to examine olfactory interactions between two canine robots. The site provides a look at several more of his concepts, including "Android Birthday" — a 13 min, 40 sec video in which a human pretending to be a robot pretends to be trying to blow out candles on a birthday cake but is unsuccessful because — after all — robots have no breath.

And I used to wonder why my BFA never led to steady employment. **SV**

Kevin Grennan's prototype robot armpit.



GEEKHEAD

by David Geer

Contact the author at geercom@windstream.net

Matt Bunting's Hexapod Robot — Takes One and Two

Intel Asks Matt To Build A Second Hexapod Design Just For Them!

It is never too soon to make unique contributions to the field of robotics. College was a perfect time for Matt Bunting, electrical and computer engineering student and roboticist at the University of Arizona. Matt mapped out and built a robust hexapod design that learns to move forward on its own, among many other interesting tasks and capabilities.

His hexapod earned him the Student of the Year award at the Annual Creativity in Electronics awards event held in Palo Alto, CA back on May 4th of this year. Matt also received and fulfilled a request from Intel to build a similar hexapod to demonstrate its fit-PC2 and Atom processor.

Matt's first iteration of the hexapod — of which there are now two — was a modestly priced configuration based on cheaper motors. The first robot used six RX-28 Dynamixel servo motors, six RX-10s, and nine AX-12s. While the configuration kept the price of this robot down, it made it difficult for Matt to readily apply an FTDI-based RS-485 adapter to communicate with the motors as he did on the second robot. This is because the motor configuration required half-duplex TTL busses.

The first robot also used foot pressure sensors to retrieve sensory information from the feet. Matt built a controller board for this using the same PIC18F4550 he had used in one of his classes, which came with an onboard slave USB 1.1 hardware peripheral and code with which to fully leverage the USB communications.

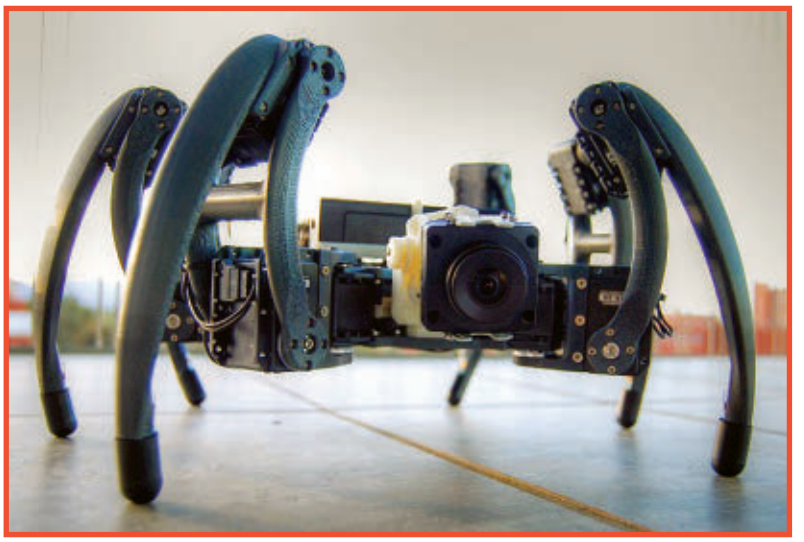
The second robot — built especially for Intel to demo what its fit-PC2 and Atom processor could do — used the FTDI-based RS-485 adapter to communicate with all of the hexapod's motors, which include 18 Dynamixel RX-28s used on the legs and three DX-117s used for the three degree-of-freedom camera operation. The legs were fabricated using a 3D printer from Stratasys.

The robot's degrees of freedom are nested in its motor locations. With 21 servo motors, the robot has 21 degrees of freedom with each leg having three degrees of freedom. "One motorized joint permits horizontal positioning of the foot, and the other two enable vertical positioning of the foot," says Bunting. The remaining degrees of freedom are in the head mechanism, enabling pan, tilt, and twist motions for the vision camera.

Matt Bunting's six-legged robot with camera and hardware, basking in the sunshine.

Learning To Move Forward

Matt used a basic reinforcement learning technique


Front angle of hexapod, in the wild and staring right at you!

called Q Learning to help the robot educate itself to walk straight forward. Using a 2D Q matrix, there are rows that represent the robot's current state, and columns that represent the possible state it could transition to. The robot is in one row and randomly chooses its next state. The robot records sensory information that tells it how well it performed on the task of moving straight forward. If it did well, it receives a high Q value which is a reward for learning well. Then, its new column is its new state and the process begins again.

As for the sensory information, this is mostly visual. The vision computation for the Q Learning requires that the robot calculate the optic flow between two successively taken shots of the same image before and after it moves. If the robot moves straight forward, the vector points in the vector field should all move outward in the second image. The robot achieves higher Q values for moving straight forward, and lower ones for moving other than straight forward. The robot uses algorithms both to determine which are high and low value movements and to decide whether to use a recorded high value movement or to try to duplicate or better a previous high value movement. Records of high value movements increase the robot's performance of forward motion over time as the number of records increases.

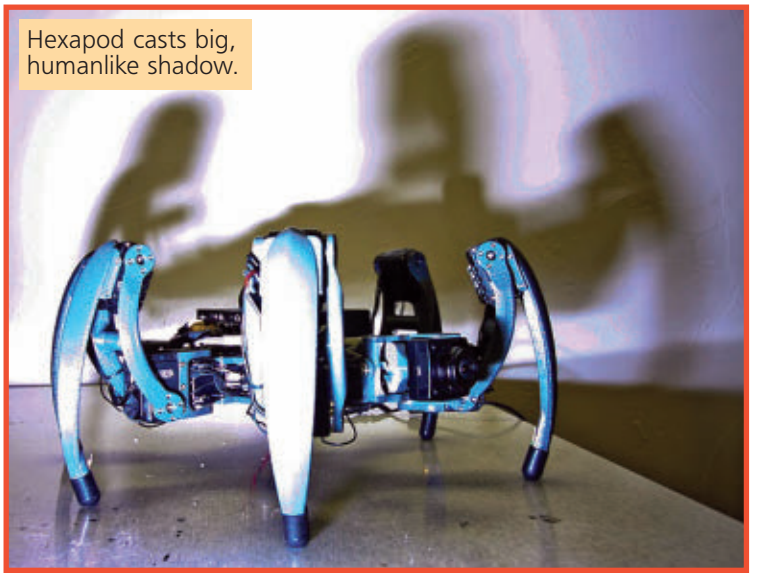
The vision programming uses OpenCV, which stands for Open Computer Vision. OpenCV is a library of vision computation functions and related functions used to make vision computations easier to implement.

For the Intel robot version, Matt used an OpenCV function to detect faces in the camera image the robot has taken. This outputs the location of the center of the face in the image. "I wrote a small program to move the camera in such a way as to center the largest face in the image," says Matt. This makes it appear that the robot is interacting with people.

Shake Your Body, Hexapod

Because Matt used a more traditional approach to operating the hexapod robot known as inverse kinematics (IK), the stepping time as the robot steps through its various gaits is very rigid, repetitive, and rhythmical, so it is often interpreted as coordinating well with music, i.e., dancing. "This is different from any other program that uses a learning algorithm," comments Bunting.

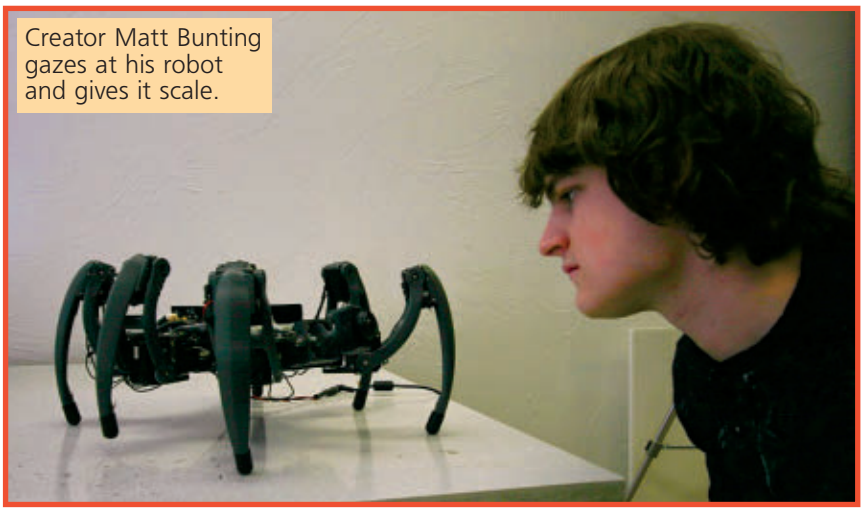
IK enables the robot to position its feet each in a specific X, Y, Z location using trigonometry to solve for the angles the motors should move into to get to those specific sets of X, Y, Z coordinates. "You have to measure all the distances of all the linkages of the legs, and determine how to set the angles of the servo motors in order to accomplish this," Matt explains.


Hexapod casts big, humanlike shadow.

The fit-PC2

The fit-PC2 from Intel — which applies the Intel Atom processor — enabled the second hexapod design with both power and computational efficiency. "The Atom processor can actually run without a heatsink while decompressing and displaying high definition video," says Matt Bunting, now University of Arizona PhD student in electrical and computer engineering. The paired Poulsbu chipset found on the fit-PC2 would usually require cooling, but not in this application.

"It is actually impressive to be able to run a full blown Ubuntu installation on a tiny 1.6 GHz machine with kinematic and vision processing code, while only using five watts of power," Bunting affirms. "It is quite an impressive little machine."


Creator Matt Bunting gazes at his robot and gives it scale.

All the IK software was coded by Matt and runs on Ubuntu Linux. "I run the code through an SSH client on either my phone or a computer," Matt continues. The programming language is C++.

The IK programming includes 3D balance gestures so the robot can balance itself and move into balanced positions as it moves. Matt explains the importance of this feature and how he derived it: "If you see a coin and try to pick it up, you don't simply extend your arm toward the coin. Rather, you bend and twist nearly all your joints so that you can comfortably perform the task. I decided to implement this approach in the hexapod to make it appear more natural and to allow ultimate flexibility of all the joints."

"I did this by imagining that real springs are attached to the hexapod body with the other end of each spring attached at the foot location. Once the feet are spread apart, the body would find the balanced position to minimize the tension of all the springs. By solving for the body location for minimum tension, the body can shift when the foot becomes too far away from or too close to the body, increasing its flexibility."

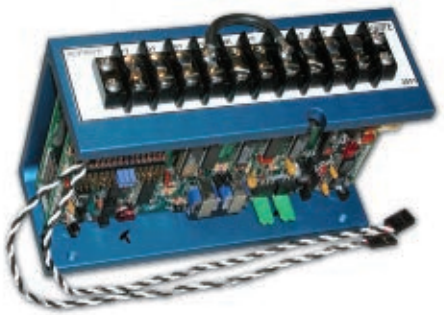
The first version of the robot performed terrain adaptation where — using IK — it employed the camera and optic flow calculations to gain knowledge of its environment. Any image that appears larger in successive images has a vector. The hexapod can use this information to perform obstacle avoidance or step up onto obstacles while keeping its body level.

In addition, this capability is a building block in autonomous navigation, environmental mapping, and exploring in which the robot uses vision information to control its legs in certain ways, to build 3D maps of the world surrounding it, and to navigate and explore in it.

Conclusion

Don't assume that familiar arenas in robotics design leave no room for discovery, innovation, and adaptation. Even a basic hexapod can fool you! SV

STEER WINNING ROBOTS WITHOUT SERVOS!



Perform proportional speed, direction, and steering with only two Radio/Control channels for vehicles using two separate brush-type electric motors mounted right and left with our **mixing RDFR dual speed control**. Used in many successful competitive robots. Single joystick operation: up goes straight ahead, down is reverse. Pure right or left twirls vehicle as motors turn opposite directions. In between stick positions completely proportional. Plugs in like a servo to your Futaba, JR, Hitec, or similar radio. Compatible with gyro steering stabilization. Various volt and amp sizes available. The RDFR47E 55V 75A per motor unit pictured above. **www.vantec.com**

VANTEC **Order at (888) 929-5055**

Resources

Matt Bunting, the robot's designer
www.engineering.arizona.edu/news/story.php?id=292

Cool robot design
www.engineering.arizona.edu/news/story.php?id=86

Dynamixel servos
http://www.robotis.com/xe/dynamixel_en

FTDI
www.ftdichip.com

PIC microcontrollers
www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2551

Stratasys 3D printers
www.stratasys.com

OpenCV library
http://sourceforge.net/projects/opencvlibrary

fit-PC2
www.fit-pc.com/web/fit-pc2/

Ubuntu Linux
www.ubuntu.com

Lynxmotion

The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!

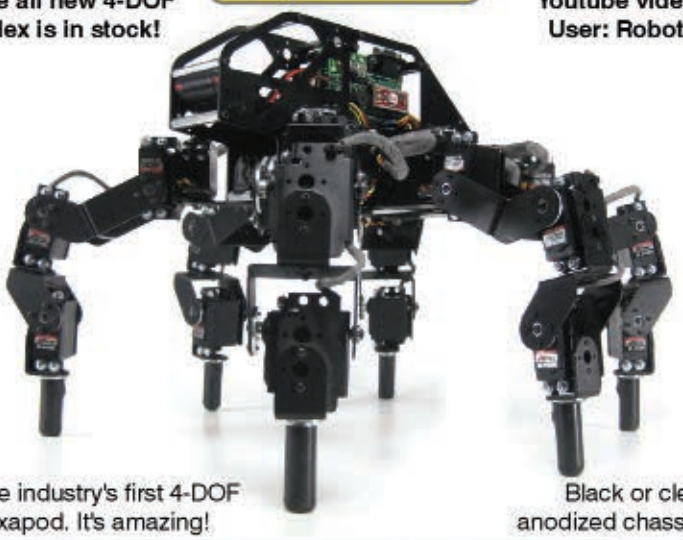
Biped Nick

Biped Pete

Featured Robot

The all new 4-DOF T-Hex is in stock!

Youtube videos User: Robots7



The industry's first 4-DOF hexapod. It's amazing!

Black or clear anodized chassis!

Biped Scout

Biped 209

Walking Stick

With our popular Servo Erector Set you can easily build and control the robot of your dreams!

Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.

All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.

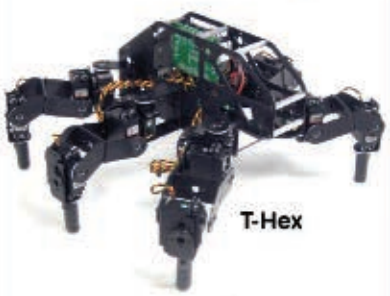
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.

SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!

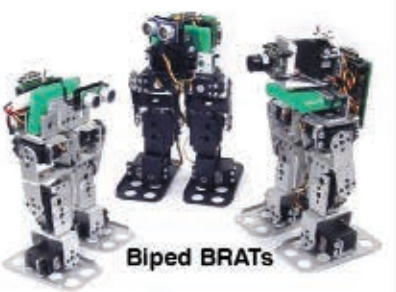
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!

Visit our huge website to see our complete line of robots, electronics, and mechanical components.

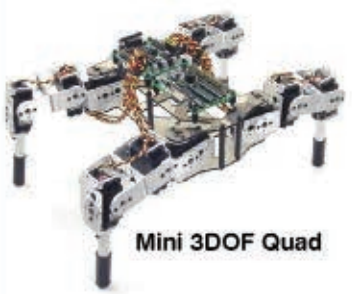
CH3-R



T-Hex



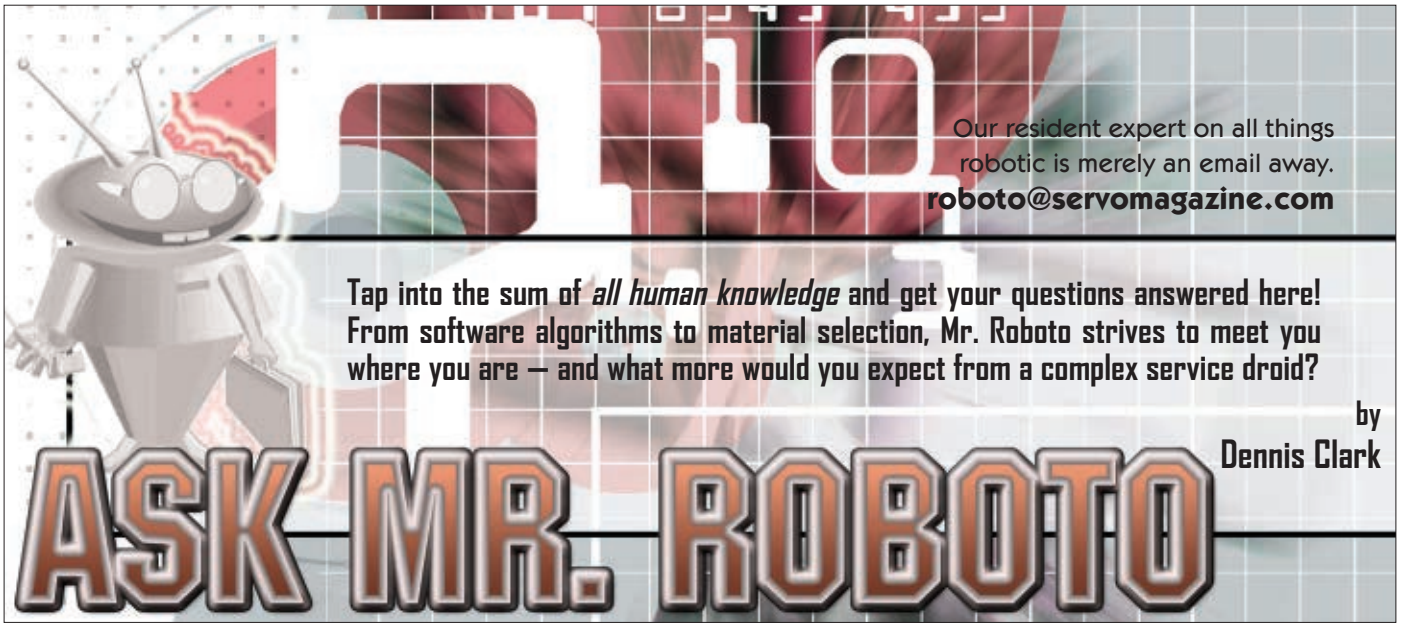
Biped BRATs



Mini 3DOF Quad

Phoenix

Images represent a fraction of what can be made! www.lynxmotion.com The SES now has over 200 unique components!



As the summer winds down (sigh), I always start to get more interesting questions. One this month about robot lawn mowers seems more in tune with the summer than the fall, but (in some parts of the USA) I guess stuff grows all year long (not in Colorado I might add, unless you are a pine tree). It appears that some of you like to keep LOTS of data on your robot from one question I got, which is kind of cool — a sort of running diary for your robot, if you are into that kind of thing! I'll say more on these things later.

As hobbyists, we have been doing line followers, wall bumpers, and mini Sumo and their ilk for quite a while now. While they do entertain the myriad gawkers at our shows and demos, they have ceased to entertain us any more. How about folks doing robot "art," like talking heads that track motion, or voice-controlled helper bots carrying your toolbox or ... In other words, are folks out there doing useful things with their robots?

I've wanted to do a robot lawn mower for a while now (one of my questions this month). I did recently get a Roomba to vacuum my lab space for me (I hate to clean) — thank you, Woot! I may not know the answer to every question that I get, but I like to research those questions and hopefully come up with answers, or even options to investigate.

We should step up to the plate as robot hobbyists, and do more than table-top conversation pieces if we want to advance the hobby. So, I challenge folks out there to knuckle their chins, scratch their heads, and generally get creative to come up with more ubiquitous robots to let loose into the wilds of our houses or yards, to actually do something that we find useful or more widely entertaining than say, a remote control mayhem monster or a table-top line follower.

The principles that we learned with our line followers and mini Sumo robots do have a use in this new world of robots. It was not wasted time! The basic idea behind the line follower can be used in our robot lawn mower to have it, for instance, follow a buried cable that it can track. My Roomba vacuum has a wall sensor (maze solver), cliff detectors (mini Sumo), and a bumper sensor. With these and the IR dome on the top, it does a pretty good job of cleaning a room.

What have you learned that you can build into a larger and more useful robot? Let's see, shall we?

Q. I am making an RC project that involves turning a push mower into a mini-zero turn type mower. I would like to know what type of motor to use. I need to know how much torque is needed to push a certain amount of weight. Should I try a high torque, low speed type motor for my project?

— Mark G.
Powell, OH

A. I have a suggestion for your platform. I've seen the "zero radius" turn mowers that the professionals use; they appear to have differential drive wheels in the back and a pair of large castors on the front of the mowing deck. That seems to be how you should build your mower.

There are some rules of thumb that you can use, or you can use math. Let's start with math. We need to know how heavy your mower platform is to start with, so we know what kind of power we need for the drive motors. I'm going to give you the gist of Chapter 2 of *Building Robot Drive Trains* that I wrote with Michael Owings a few years ago. This is a thumbnail discussion of how you can estimate what you'll need for motors. If you are interested in going into more depth, check this book out of your local library, or, buy a copy of your very own (for which Michael and I will thank you).

Even though this discussion will get into some math, I promise, it won't hurt a bit! First off, we want to estimate what power we'll need in our motors to move the mower the way you want it to move. To move our robot, we'll be working against two basic forces: friction and gravity. Together, we get this:

$$F_{app} = F_f + F_w$$

The force we need to apply () must be greater than the force of friction plus the force of weight (due to gravity) to move. The force due to gravity is shown as this formula:

$$F_w = mg\sin\theta$$

which means the product of our mass and acceleration due to gravity times the sin of the angle from the perpendicular to the ground is the force due to gravity that needs to be overcome. As obvious as this formula may seem (that was a joke, really), it may need some explanation.

We all get the "heavy things are hard to move" part, but what is that *sin theta* thing all about? It's easy, really. Push something on level ground, grunt a bit, and off it goes, right? Well, now push it up your driveway at a 20 degree angle. Suddenly it isn't so easy to push, right? Now, for the next bit, friction — the other half of our simple equation above. This is the formula for the force to overcome friction:

$$F_f = \mu mg\cos\theta$$

The first term, *mu*, is the coefficient of friction — *static* friction (more on that later.) You have seen "mg" before; it is weight — the cosine is of the angle off of the horizontal plane. What this all comes down to is that friction is highest on flat ground and zero when falling straight down; that makes sense. There are two kinds of friction: static friction, which is the friction to overcome to get moving; and dynamic friction — that which needs to be overcome while you are in motion. The latter is far less than the former. Even a wheel has a surface area in contact with the ground; it isn't just a single point, it has width and length. Until that wheel starts to turn, it is basically just skidding on the ground.

Once the wheel starts turning though, everything changes and the friction goes way down. Push anything with wheels. Did you notice that you really had to heave to get it going? Once it is moving, it is much easier to keep moving. That is partially friction; the other part is Newtonian physics — momentum; objects in motion tend to want to stay in motion.

Back to the point. The coefficient of friction is rarely given, but can be measured. If you are interested, it is the ratio between the *resistive* friction over the *normal* friction. Normal friction is basically the force holding things together, *mg*, or weight. To get the resistive friction, get a fish scale and attach it to the object and pull. That force will look like weight since this is a scale; take that value. Simple.

Now, we have the final formula for measuring the force needed to move your robot lawn mower:

$$F_{app} = mg\sin\theta + \mu mg\cos\theta$$

I like to work in metric units, so g = 9.8m/s^2, therefore, I use mass in kilograms. Finally, we need power. This formula is:

$$P = F_{app}v$$

which means power is force times velocity. For that to be useful, we need this formula for motors:

$$\omega = v/r \Rightarrow v = \omega r$$

Rotational velocity is velocity divided by the radius of the arm from the center of the motor. The second formula just puts it into the form we want for the power formula above.

Now for units. These are the units to work with when using these formulas to get the power you need for your motors:

Term	Unit
Power	Watts (W)
Force	Newton Meters (Nm)
Mass	Kilograms (kg)
Angle	Radians
Rotational Velocity	Radians/Sec

Pick the angle to match the inclines that will most likely be the worst case scenarios in your yard. Pick a coefficient of friction to be something like a car tire, which I've found to be about .9 to 1.0 for a rubber tire on concrete (to be conservative). If you choose 1.0, obviously that term falls out. Just plug in your numbers.

How do you know the power of your motors? Sadly, no one ever gives you that. If you get a motor brand new from a manufacturer, you might be able to get the maximum torque and maximum rotational velocity, which gives you the power by this formula:

$$Pm = T\omega$$

The motor power is the torque times the rotational velocity; this gives you the power at any chosen torque and velocity, but not the maximum power. The maximum torque in a DC motor is at zero rotational velocity, which is no power. The maximum rotational velocity is where the motor torque is at its minimum. To get a motor's maximum power, it needs to be where the torque is at 1/2 and the velocity is at 1/2, which gives this formula:

$$P_{\max} = \frac{1}{4}T_{\max}\omega_{\max}$$

Torque is force and (here) is shown as Nm; rotational velocity in is radians/second. I don't know the English units; I'm more comfortable in metric. Torque is the angular force that a motor can deliver at some distance from the shaft. If your motor could lift 1 kg on a pulley with a one meter radius, that would be one Newton meter. There are 2p radians in a full circle.

This suggests a way to get the power of your motor empirically; lift weights using a measured pulley, or pull on

your fish scale if you have a big motor.

That is the math you would use to choose your motors. This will get you close to what you need for success. Gordon McComb (of *Robot Builders Bonanza* fame) shared his secret for success: the heft method. He would look at his robot frame, pick up a motor, and hold it to gauge its *heft* to determine its suitability. I myself do not have Gordon's calibrated arm, so I tend to do a little math.

Q. I want to store way-points and sensor data during a robot's run-time. This really adds up and is way more than most EEPROMs I've seen can do. I've been trying to get an SD card to work using an SPI interface, but it just isn't working. I can't initialize the card. How does this work?

— Thomas Q.
Boise, ID

A. I have been fascinated with SD cards for a variety of reasons, and this gives me a great reason to work with them. I am currently having fun with Microchip PIC24 devices and so used one of their demo boards with an SD PICtail card for experimenting. The target hardware isn't all that important, so my code should work with anyone's microcontroller; just change how you deal with the SPI hardware to get your required clock rates. An SD card can take SPI clocks of 20 to 50 MHz, so there is no issue about going too fast. As it happens, my 32 MHz PIC24FJ64 part can only go 8 MHz, but that gave me pretty good transfer rates (compared to serial ports, anyway) and I was happy. The secret to success is to start out at a clock rate of 400 kHz until the card is up and listening, then move to the high speed clock. To fully understand the interface — which is pretty simple — and the protocol — which is slightly less simple, check out this site and get the SD Association *Simplified* Specs: **www.sdcard.org/dev.../pls/simplified_specs**.

This site took me a long way towards getting my SD cards to work: **http://elm-chan.org/docs/mmc/mmc_e.html**.

There is a raft of information out on the net for handling the reading and writing of data blocks to the SD card, so that part was easy. The big thing to know is that the SD card wants the block address in bytes, on 512 byte boundaries; the SDHC card has a much higher capacity and wants its block addressing done where each block is 512 bytes. My initialization code shown here takes this into account by storing the attributes of the SD or SDHC card being used so that the read/write routines know which type of block addressing to use. Many thanks to the pioneers of SD card usage that helped me with this work!

I have given my various routines here, but did not bother with the actual block read and writes. I'll leave that as an exercise for the reader. If anyone is interested, you can send me an email (see end of article) and I'll do a more detailed write-up.

There are a lot of constants in **Listing 1**. They refer to various values for commands, which are pretty obvious when you look at the specifications for SD commands. I

Listing 1: SD Card Initialization.

```
uint8_t SD_Write(uint8_t b)
 /**
 *  Read and write a single 8-bit word to
 *  the SD/MMC card. Using standard, non-buffered
 *  mode in 8 bit words.
 *  **Always check SPI1RBF bit before reading
 *  the SPI2BUF register
 *  **SPI1BUF is read and/or written to
 *  receive/send data
 *
 *  PRECONDITION: SPI bus configured, SD card
 *  selected and ready.
 *  INPUTS: b = byte to transmit (or dummy
 *  byte if only a read done)
 *  OUTPUTS: none
 *  RETURNS:
 */
 {
  SPI1BUF = b;
         // write to buffer for TX
  while( !SPI1STATbits.SPIRBF);
         // wait for transfer to complete
  SPI2STATbits.SPIROV = 0;
         // clear any overflow.

  return SPI1BUF;
         // read the received value
 }

 // Not worth code defining these since
 // they are all the same.
 #define SD_Read()    SD_Write( 0xFF)
 #define SD_Clock()    SD_Write( 0xFF)
 #define SD_Disable() nMEM_CS = 1; SD_Clock()
 #define SD_Enable()  nMEM_CS = 0

 uint8_t SD_SendCmd(uint8_t cmd, LBA addr)
 /**
 * Send an SPI mode command to the SD card.
 *
 * PRECONDITION: SD card powered up, CRC7
 * table initialized.
 * INPUTS: cmd = SPI mode command to send
 *          addr= 32bit address
 * OUTPUTS: none
 * RETURNS: status read back from SD card (0xFF
 * is fault)
 * *** NOTE nMEM_CS is still low when this
 * function exits.
 *
 *  expected return responses:
 * FF - timeout
 * 00 - command accepted
 * 01 - command received, card in idle state
 * after RESET
 *
 *  R1 response codes:
 * bit 0 = Idle state
 * bit 1 = Erase Reset
 * bit 2 = Illegal command
 * bit 3 = Communication CRC error
 * bit 4 = Erase sequence error
 * bit 5 = Address error
 * bit 6 = Parameter error
 * bit 7 = Always 0
 */
 {
   uint16_t     n;
   uint8_t        res;
   uint8_t        byte;
   uint8_t        CRC7 = 0x95;
         // Generic CRC7 byte

   SD_Enable();
         // enable SD card
```

```
  byte = cmd | 0x40;
  SD_Write(byte);
        // send command packet (6 bytes)
byte = addr>>24;
  SD_Write(byte);
        // msb of the address
  byte = addr>>16;
  SD_Write(byte);
  byte = addr>>8;
  SD_Write(byte);
  SD_Write( addr);
        // lsb

  SD_Write(CRC7);
        // Not used unless CRC mode on

  // now wait for a response (allow for up
  // to 8 bytes delay)
  n = 9;
  do {
    res = SD_Read();
        // check if ready
    if ( res != 0xFF)
        break;
  } while ( —n > 0);

  return (res);
        // return the result
}

void SD_InitSPI( void)
/**
*   Configure the SD card SPI bus hardware
*   settings and software. *
*** Using the SD SPI mode spec settings
*   instead of the MCHP example.
*
*   PRECONDITION: none
*   INPUTS: none - The hardware is explicitly
*   set up
*   OUTPUTS: none
*   RETURNS: none.
*/
{
  nMEM_CS = 1;
        // De-select the SD card

  if (sdcard.cardInit == 1) {
      return;
  }
    // init spi module for a slow (init) clock
    // speed, 8 bit byte mode
    // Master, CKE=0; CKP=1, sample end,
    // prescale 1:64 (250KHz)
  SPI1STATbits.SPIEN = 0;
        // disable SPI for configuration
    SPI1CON1 = 0x027c;
    SPI1CON2 = 0x0000;
        // No buffer, no frame mode
    SPI1STAT = 0x8000;
        // enable
}

uint8_t SD_InitMedia( void)
/**
*  Discover the type and version of the
*  installed SD card.  This routine will find
*  any SD or SDHC card and properly set it up.
*
*  PRECONDITION: none
*  INPUTS: none
*  OUTPUTS: none
*  RETURNS: 0 if successful, some other
*  error if not.
*/
{
  uint16_t     n;
  uint8_t     res = 0;
  // // If we get that far...
  uint32_t    timer;
  uint8_t        cmd;
  uint8_t        db[16];
    // when we get data back to look at

  if (sdcard.cardInit == 1) {
      return(0);
    // done, don't do it again.
  }

  // 1. start with the card not selected
  SD_Disable();
  // 2. send 80 clock cycles so card can
  // init registers
  for ( n=0; n<10; n++)
      SD_Clock();
  // 3. now select the card
  SD_Enable();

  // 4. send a reset command and look for
  // "IDLE"
  res = SD_SendCmd( RESET, 0); SD_Disable();
  if ( res != 1) {
     SD_Disable();
     return(LOG_FAIL);
        // card did not respond with "idle"
    }

  // 5. Check card voltage (type) for SD 1.0
  // or SD 2.0
  res = SD_SendCmd(SEND_IF_COND, 0x000001AA);
  // didn't respond or responded with an
  // "illegal cmd"
  if ( (res == 0xFF) || (res == 0x05)) {
      sdcard.cardVer = 1;
        // means it's an SD 1.0 or MMC card

    // 6. send INIT until receive a 0 or
   // 300ms passes
    timer = t_1ms + 300;
     while(timer > t_1ms) {
        res = SD_SendCmd(INIT,0);
        SD_Disable();
        // SendSDCmd() enables SD card
        if (!res) {
            break;
        // The card is ready
        }
    }
    if (res != 0) {
        return(LOG_FAIL);
        // failed to reset.
    }
    SD_Disable();
        // remember to disable the card
 }
else {  // need to pick up 4 bytes for v2
    card voltage description
        sdcard.cardVer = 2;
        // SD version 2.0 card
        for (n=0; n<4; n++) {
            db[n] = SD_Read();
        }
        // but we'll ignore these bytes
        SD_Disable();

       // 6. send INIT or SEND_APP_OP
       // repeatedly until receive a 0
       cmd = SEND_APP_OP;
       timer = t_1ms + 300;
        // wait up to .3 seconds for life
        // will still be in idle mode (0x01)
        // after this
       res = SD_SendCmd(APP_CMD, 0);
        SD_Disable();
       while (timer > t_1ms) {
        res = SD_SendCmd(cmd, 0x40000000);
       SD_Disable();
       // ACMD41 not recognized, use CMD1
```

Cont.

Firgelli
www.firgelli.com
LINEAR SERVOS
L12-R Linear Servo
• Direct replacement for regular rotary servos
• Standard 3 wire connectors
• Compatible with most R/C receivers
• 1-2ms PWM control signal, 6v power
• 1", 2" and 4" strokes
• 3-10 lbs. force range
• 1/4" to 1" per second speed ranges
• Compatible with VEX
L16 Linear Actuators
New!
• 2", 4" and 6" strokes
• 10 - 40 lbs. force range
• 1/2" to 1" per second speed ranges
• Options include Limit Switches and Position Feedback
PQ12 Linear Actuator
• Miniature Linear Motion Devices
• 6 or 12 volts, 3/4" stroke
• Up to 5 lbs. force
• Integrated position feedback or limit switches at end of stroke
• External position control available
Linear Actuator Controller (LAC)
• Will drive any Linear Actuator with position feedback
• Up to 24v and 4 Amps
• USB connectivity to drive the actuaor with your computer
• Adjustable speed, limits and sensitivity
L12-NXT Linear Servo
• Designed for LEGO Mindstorms NXT®
• Plugs directly into your NXT Brick
• NXT-G Block available for download
• Can be used with Technic and PF
• Max. speed: 1/2" per sec.
• Pushes up to 5 lbs.
• 2" and 4" strokes
Available Now @
www.firgelli.com

```
        if ( (res &0x0F) ==
       0x05 ) {
              cmd = INIT;
        }
        else {
            cmd = SEND_APP_OP;
        }
       if (!res) {
            break;
       }
     }
     if (res != 0) {
         return(LOG_FAIL);
        // failed to reset.
     }

     // 7. Check for
     // capacity of the card
     res = SD_SendCmd
    (READ_OCR,0);
     if (res != 0) {
       return(LOG_FAIL);
        // error, bad thing.
     }
     for (n=0; n<4; n++) {
       db[n] = SD_Read();
     }
     SD_Disable();
    // check CCS bit (bit 30),
    // PoweredUp (bit 31) set
    // if ready.
     if ( ((db[0] & 0x40)
    == 0x40) && (db[0] !=
    0xFF) ) {
         sdcard.cardCap = 1;
      // card is high capacity
      }
      else{
         sdcard.cardCap = 0;
      // card is low capacity
      }
}

sdcard.cardInit = 1;
   // successfully initialized

// Get the CSD register to
// find the size of the card
res = SD_SendCmd(SEND_CSD,0);
if (res != 0) {
    return(LOG_FAIL);
}
timer = t_1ms + 300;
       // wait for a response
while(timer > t_1ms) {
    res = SD_Read();
    if (res == DATA_START) {
        break;
  }
}
if (res == DATA_START) {
    // no timeout, read data
     for (n=0; n< 16; n++) {
         db[n] = SD_Read();
    // read the received value
     }
     // ignore CRC (for now)
        SD_Read();
        SD_Read();
        SD_Disable();
    }
    else {
        return(LOG_FAIL);
    }
    if (sdcard.cardCap == 1) {
         // Uses SDHC capacity
         // calculation
         sdcard.cardSize =
         db[9] + 1;
         sdcard.cardSize +=
        (uint32_t)(db[8] << 8);
         sdcard.cardSize +=
        (uint32_t)(db[7] &
         0x0F)<<12;
         sdcard.cardSize *=
         524288;
         // multiply by 512KB
         // (C_SIZE + 1)
         // * 512 * 1024
         sdcard.cardNumBlocks =
         sdcard.cardSize/sdcard
         .cardBlock;
    }
    else {
         // Uses SD capacity
         // calculation
      sdcard.cardSize =
      (uint16_t)((db[6] &
      0x03)<<10) | (uint16_t)
      (db[7]<<2) | (uint16_t)
      ((db[8] & 0xC0)>>6)) + 1;
       sdcard.cardSize =
      sdcard.cardSize <<((
      (uint16_t)((db[9] &
      0x03)<<1) | (uint16_t)
      ((db[10] & 0x80)>>7)) +2);
       sdcard.cardSize =
      sdcard.cardSize <<(
      (uint16_t)(db[5] & 0x0F));
      // (C_SIZE +1) <<(C_SIZE_
      // MULT + 2) <<(READ_BL_LEN)
      sdcard.cardNumBlocks =
      sdcard.cardSize/
      sdcard.cardBlock;
// Set block size to 512 bytes
      res = SD_SendCmd(
       SET_WBLEN,0x00000200);
      SD_Disable();
    }

    // Now kick to full speed
    // 8MHz mode. disable SPI for
    // configuration
    SPI1STATbits.SPIEN = 0;
    // Master, CKE=0; CKP=1,
    // sample end, prescale 1:2
    // (8MHz) all works
    SPI1CON1 = 0x027b;
    // re-enable SPI after
    // configuration
    SPI1STATbits.SPIEN = 1;
    return(res);
}
```

don't like "magic numbers" in my code, so I use defined constants. This procedure will identify both SD and SDHC cards. I have not tested it with SDXC cards, however, since I don't have any of them.

Whew! Another Mr. Roboto has come to an end. There were many obstacles that got in my way to get here, but "where there is a will ..." If you have any questions for Mr. Roboto, please send them to roboto@servomagazine.com. I love to hear from you and will do my best to answer your questions. Until then, keep on working on those robots! **SV**

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

SEPTEMBER

2-5 **DragonCon Robot Battles**
Atlanta, GA
Remoted controlled machines destroy each other as entertainment for science fiction fans and cosplayers.
www.dragoncon.org

5-10 **National Junior Robotics Competition**
Science Centre, Singapore
Student teams develop problem-solving skills creative thinking, and team spirit as they build autonomous robots that compete for the championship award.
www.science.edu.sg/events/Pages/njrcompetition.aspx

12-15 **International Micro Air Vehicle Competition**
't Harde, The Netherlands
Indoor and outdoor events for tiny flying robots. Also, a special contest for Parrot AR Drones which are relatively huge compared to the typical MAV. Parrot is sponsoring this year with free loners of Parrot AR Drones to participating teams. All events are for fully autonomous robots only; no remote control allowed except for tech demonstrations.
http://www.imav2011.org/

17 **Robotour**
Vienna, Austria
Autonomous robots must navigate through a public park, carrying a payload that consists of a five liter barrel. All robots receive a map and destination coordinates prior to the race, and all bots will start concurrently.
www.robotika.cz/

24-25 **Robocup Junior Australia**
University of Tasmania, Australia
Student teams build autonomous robots that compete in three events: soccer, rescue, and dance.
www.robocupjunior.org.au

OCTOBER

20-21 **Competencia Robotica (LARC)**
Center for Robotics UTFSM, Valparaiso, Chile
This year's events are Robot Freighter and Police Robot. Robot Freighter is a way-finding contest in which robots must pick up a payload and take it to a destination. Police Robot is a contest where autonomous robots must surveil a city, locate criminals, and deploy guard robots in areas with too many criminals.
http://robotica.elo.utfsm.cl/competencia/

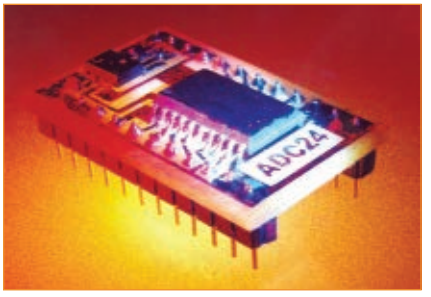
21-22 **CalGames**
Archbishop Mitty High School, San Jose, CA
This is a FIRST-based robotics event for high school teams.
www.wrrf.org

21-23 **Critter Crunch**
Hyatt Regency Tech Center, Denver, CO
Autonomous and remote-controlled robots, and other machines battle each other in hopes of winning prizes that include handmade objects by local artists and awards for "amusing and arbitrary achievements."
www.milehicon.org/?page_id=16

28-30 **COMBOTS Cup**
San Mateo, CA
Remote-control vehicles destroy each other.
http://combots.net

NEW PRODUCTS

Hybrid A/D Conversion "System on a Chip"



Elk Industries, LLC announces the release of its new ADC24 Analog-to-Digital System CMOS Hybrid Integrated Circuit. The unique features of this product make an easy to use analog-to-digital conversion system, all in a 24-pin DIP IC. Smaller in size and weight, this A/D conversion system requires only +5 volts, DC, and ground, as well as the analog signal input. It instantly produces eight-bit parallel digital data on its outputs.

This "system on a chip" can handle analog signals from DC to 60-100 kHz in bandwidth. Only valid datum is presented on the (high true) digital data outputs. A convenient SYNC output indicates the latest update of the digital data on the outputs. The entire system fits into a standard 24-pin DIP IC socket and comes complete, ready to use.

Some of the features include single +5 volt operation, miniature size and weight, full eight-bit resolution, and its low power consumption (<100 milliwatts, typically). Each Hybrid IC is 100% performance tested. Applications include industrial controls, data acquisition systems, instrumentation, digital audio, communications, and automation, just to name a few. With the ever-increasing need to "digitize" data and signals in the real world, the implications of this advantageous system on a chip are several. The single IC approach makes A/D conversion easier because by using this device, essential real estate on circuit boards is reduced substantially, and is accomplished without increasing overall power consumption.

By using this IC solution, overall system costs can be reduced, as well. A detailed datasheet is provided to answer questions relating to hook-up and other design considerations. The advanced analog-to-digital conversion system technology is part of a New Transformerless Battery Charging Technology developed by American Advanced Technologies, Inc. (AAT), and is tailored for use in the electric vehicle industry.

For further information, please contact:

Elk Industries, LLC
221 W. Hibiscus Blvd. #106
Melbourne, FL 32901
321 • 259 • 8114 Fax: **321 • 723 • 2739**
Email: **laserob@aol.com**
Website: **www.elkindustries.com**

New PROFINET and PROFIBUS Rotary Encoders for Automation Technology

To help keep up with the growing demands of the industry, Heidenhain has increased the functionality of its PROFIBUS rotary encoders. The rotary encoders with a PROFIBUS interface now support the DP-V2 encoder profile which makes them ready for robot control and production technology.

The standard rotary encoders of Heidenhain's ROC/ROQ 400 series with 58 mm housing diameter are now available with the PROFINET interface. These encoders can be used in manufacturing systems, process automation, and drive applications.

Within these newly improved encoders are M12 or M23 input connectors and flange sockets that make cable connections more accessible and easier to use. M12 flange sockets allow the use of preassembled standard cables which helps avoid incorrect wiring, while reducing the number of cables assembled in the field.

This connection technology provides a further advantage during commissioning: It reduces the risk of ESD damage to the encoder. Absolute rotary encoders with PROFIBUS comprise single-turn and multi-turn versions. They can be configured and parameterized according to the requirements of the application.

Versions with the PROFINET interface are new to the Heidenhain rotary encoder program. These encoders support the functions of Conformance Class 4 (full scaling and preset functions) and offer ETHERNET-based real time and IT communication.

For further information, please contact:

Heidenhain Corporation
333 E. State Parkway
Schaumburg, IL 60173
Tel: **847 • 490 • 1191**
Website: **www.heidenhain.us/rotary_encoders.php**

Waterproof Servos

Hitec's new waterproof servos have what it takes to keep your radio control vehicle, airplane, or robot watertight. With high torque, dual ball bearings, and metal gears, both the analog and digital waterproof servos have the

industry's first IP67* rating, ranking them among the most durable and reliable servos for wet conditions.

Features:

- Metal gear with MP first gear.
- Dual ball bearings.
- Three-Pole cored ferrite motor.
- Industry's first IP67-rated waterproof case.*
- Two-Cell LiPo capability.
- Programmable circuit.

For further information, please contact:

Hitec

12115 Paine St.
Poway, CA 92064
Website: **www.hitecrcd.com**

Li-ion Power Pack/Charger – 2 Cell

The Li-ion Power Pack-Charger – 2 Cell is an integrated storage cell and charging system on a single 3" x 4" printed circuit board now available from Parallax. It's compatible with most 18650-size Li-ion cells. Price is $49.99.

Features:

- PCB-mounted cell holders with on-board charging circuitry.
- Multiple power input/output options.
- On-board output fuse protection.
- Nominal 7.4 VDC output; 8.2 VDC maximum.
- Standard 3" x 4" PCB footprint integrates well with the Board of Education® (#28150), Propeller™ Proto Board (#32212), or any application needing a reliable power. supply with an integrated charging system.
- Automatic charge/discharge switching circuitry.
- Holds two rechargeable 3.7 volt Li-ion 18650-size cells.
- Multiple LED indicators provide charge readiness information for each individual cell; status key for the LED indicators is printed on the board.
- Aggressive holders retain cells in any board orientation and in moderate shock environments, such as mobile robotic applications. Cells are not permanent and are easily replaced.
- Dedicated circuitry continuously monitors the charging process to ensure safety, efficiency, and to maximize the number of charge/discharge cycles of each cell.

For further information, please contact:

Parallax, Inc.

Website: **www.Parallax.com**

NEXYS3 FPGS Board

Digilent, Inc., announces the launch of the NEXYS™ 3 board, the latest addition to their family of FPGA-based design kits. Featuring Xilinx's newest Spartan6 FPGA, 48 Mbytes of external memory (including two non-volatile phase-change memories from Micron), USB and Ethernet ports, and other common I/O devices, the Nexys3 board is perfectly suited for academic and industrial design settings. Students and engineers alike can build circuits ranging from simple logic circuits, to CODECs and custom device controllers, to embedded processors without needing any other equipment and for less than the cost of a textbook.

The Nexys3's high speed USB2 port provides board power, high speed FPGA programming, and user-data transfers at rates up to 38 Mbytes/sec. The USB2 port supports Digilent's latest Adept programming software as well as all Xilinx tools, including the free WebPack™, Chipscope™, and EDK™ tools. In addition to FGPA and ROM programming, the Adept system offers automated board tests, a virtual I/O system that extends I/O capabilities using a PC-based interface panel, and simplified user-data transfers. The Adept application, an SDK, and reference materials are freely downloadable from the Digilent website.

A large collection of low cost peripheral boards — including more than 40 Pmods™ and several new Vmods™ — can add additional features to the Nexys3, including A/D and D/A converters, breadboards, motor drivers, displays, etc.

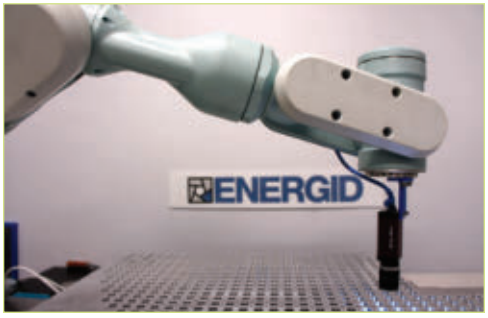
The Nexys3 board is priced at $119 for academic customers and $199 for all others. Volume discounts are available. Most orders ship the day they are received.

For further information, please contact:

Digilent, Inc.

Website: **www.digilentinc.com**

bots IN BRIEF



INSPECTOR BOT

Considering their current problems with robots trying to help in the nuclear cleanup, Japan's Mitsubishi Heavy Industries has made an agreement with Energid to develop a power plant inspection robot. The plan is to make one that can inspect the heat exchanger and steam tubing system faster and more reliably. COO Neil Tardella says he believes Energid was chosen because they have worked with NASA in developing the K10 rover.

AWESOME ATR

OWI always makes their kits simple for kids of all ages to build. They're now offering a 3-in-1 ATR that can act as a forklift, rover, and gripper. A wired controller allows the All Terrain Robot to go forward, backwards, turn, and operate with a maximum lift of 200 gm. The ATR needs four AA batteries (not included).

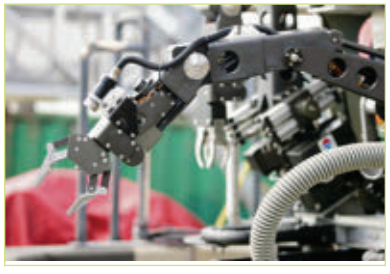
ROBOSUB CHALLENGE WINNERS

The 2011 International Autonomous Underwater Vehicle Competition — now dubbed RoboSub — took place July 12-17. The RoboSubs are autonomous (not remote controlled), so the competition operates similar to the DARPA Grand or Urban Challenge: You push the go button and then your robot is on its own; you can do nothing but sit back and have an anxiety attack. The competition was held at the US Navy's SPAWAR System Center down in San Diego, CA where nearly 30 teams (including both high school and international teams) unleashed their autonomous robot submarines against a hapless swimming pool filled with gates, buoys, paths to follow, objects to retrieve, and targets to torpedo. Winners were as follows:

1st Place: ETS Team SONIA (awarded $7,000)
2nd Place : Cornell University (awarded $4000)
3rd Place: University of Florida (awarded $3,000)
4th Place : Reykjavik University (awarded $2,000)
5th Place: University of Maryland (awarded $500)
6th Place: University of Rhode Island (awarded $500)
7th Place: United States Naval Academy
8th Place: NC State

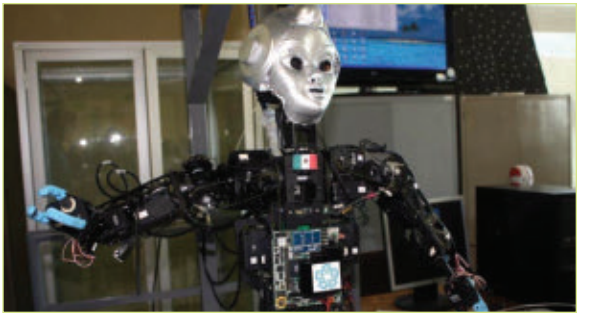
MUNITIONS MOVER

The Army is conducting a 21 day trial run with a remote controlled submersible. The Remotely Operated Underwater Munitions Recovery System (ROUMRS) robotic arm has been specifically designed to remove WW II munitions. It will be recovering grenades, bullets and their casings, and bombs. ROUMRS is similar to submersibles used by offshore oil rigs and can go down 120 ft to vacuum up small objects. After the trial run, damage to the reef, etc., will be studied and — if considered successful — the military will start the removal next year.



bots IN BRIEF

MEX-AMORPHOSIS



Mex-One — Mexico's first humanoid robot — is entering the final phase of its metamorphosis. It is a humanoid that not only can walk and memorize information, it is considered a social robot and can learn by experience. Team leader Eduardo Bayro-Corrochano claims that they will have "sub-products with practical ends in such fields as medicine and culture." This means that it has open architecture for developing new algorithms, ideas, and applications such as intelligent limbs for the disabled. The total cost is about $100,000, but now Professor Bayro-Corrochano says one could be made for $50,000. Mex-One stands 105 cm tall and weighs 15 kg.

KIBOT FOR KIDS

Korea's latest playmate for kids is a multi-tasking monkey aimed at three to seven year olds. The 8" tall wheeled Kibot can be controlled via a cellphone by parents so that kids can video chat with others. Made by KT, Kibot comes in gray or pink, responds to gestures by talking, teaches language with its 3.5" display, and avoids obstacles when necessary. The bilingual bot is available for 485,000 won (~$447) with wireless packages available. Songs, stories, and animation can be downloaded after purchase.

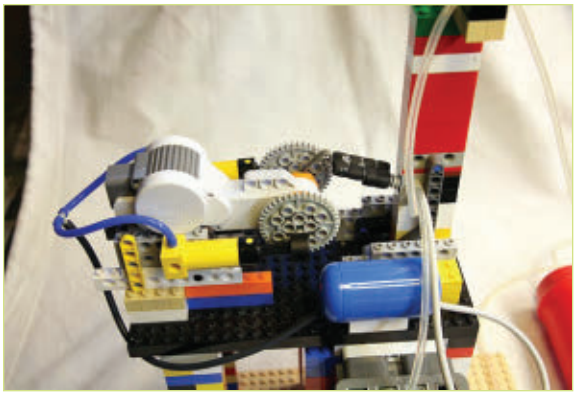
GET SCREENED

Joey Daoud, director of the film *Bots High*, has announced that there will be a free screening day on October 6th, so mark your calendar now to attend. If you didn't know already, *Bots High* follows three teams of students who are heading to a national robotics competition. There are 22 screenings planned so far, and you can suggest a location or hold your own by gathering a group of 25 and then contacting Daoud at **www.botshigh.com**.

MAY THE FORCE BE WITH YOU

A student project at Stanford University has resulted in the creation of JediBot — a robot arm that will actually try to "kill" you with a foam sword, especially since the robot applies quite a bit of "force."

This project was part of Stanford's three and a half week long "Experimental Robotics" course which — from the sound of things — is basically just an excuse for students to mess around with robots to get them to do cool stuff. Also developed as part of the course were a robot that plays golf, several robots that draw, and a robot that can make hamburgers and then drown them in ketchup for you.



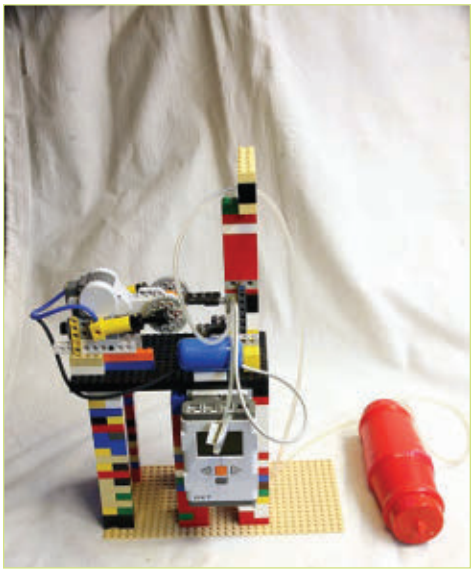
LEGO MY PANCAKE

The LEGO Pancake Bot created by Miguel Valenzuela is a work in progress, designed to make pancakes in shapes. Two empty ketchup bottles serve as dispensers on an aluminum protected track, and a three-axis Computer Numerical Control (CNC) handles the movement with Python NXT — a scripting language for LEGO projects. It uses the Z coordinate as the pancake batter dispenser control.

The Pancake Bot is made up of the following parts:

1. A set of linked base plates with three parallel tracks snapped on. One track has a bunch of Technic Gear Racks (1 x 4) riding along the top which makes up the X axis.

2. A moveable bridge that holds one NXT motor on one side and has free rolling wheels on the other. The top of the bridge uses two tracks covered with gear racks that holds the pancake Batter Dispenser Unit (BDU). This track allows for Y axis control.



3. A carrier gantry on top of the moveable bridge that rides on the gear racks and holds the BDU.

4. As mentioned, the BDU consists of two ketchup bottles cut in half and glued together so you have two open ends. This allows you to change the nozzle size on the bottom and allows for input of compressed air on top.

5. A Compressed Air Dispersal Apparatus (CADA) that consists of an NXT motor, two pneumatic cylinders, a tank, and a bi-directional flexible switch which allows for the switch to be flipped based on which direction the NXT motor turns. As the motor turns, the cam changes the flexible switch every time it rotates. In the initial turn, the switch is flipped and stops, and then continues to flex out of the way of the rotating cam. When the motor turns the other way, the cam catches the switch, flips it back, and changes the direction of the airflow. It continues to flex as the cam passes over it. This allows for air to be compressed when the motor turns clockwise, and then the air to be released when the motor turns counter-clockwise.

Valenzuela designed the bot this way because he only had three NXT motors and needed a way to change the direction of the air flow while at the same time, compressing the air. Go to **http://makermig.blogspot.com/2011/06/how-pancake-bot-works.html** for more details and videos.

DIGGER TEARS THINGS UP

The Digger D-3 is the most recent addition to my own personal list of robots not to stand in front of. It's a mine-clearing robot, and not the sort of mine-clearing robot that pokes around with a metal detector. Instead, it simply sucks it up and tells the landmines to bring it.

At the front of the D-3 is a giant spinning metal pulverizer of death which has Tungsten hammers that beat down a quarter meter into the ground, turning everything they touch into mulch. This includes landmines, and although the mines do tend to blow up before getting shredded, the robot hardly seems to notice.

An operator commands Digger from a safe distance using a remote control unit. The hull of the robot is made up of hardened steel plates in a V shape to help limit any damage from antitank mines and unexploded shells of sizes up to 81 mm. The D-3 has been able to successfully ingest mines containing as much as eight kilograms of explosive — which is nothing to sneeze at. The only potentially vulnerable spots are the air intakes which are themselves protected from flying shrapnel by special grates.

At full throttle, the D-3 can reliably clear a comforting 100 percent of landmines from the ground at a rate of 1,000 square meters per hour (about 10,000 square feet per hour), while also divesting the land of any unwanted shrubbery and unlucky mole colonies.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

WED THIS WAY

You have to love AutoWed from Concept Shed. We're talkin' instant matrimony here. Put in your dollar and it talks you through the ceremony after you input your names — music is included. The robopreacher gives you a certificate/receipt and a couple of rings that look like they came out of a Cracker Jack box. Who knows how legal it is, but it has at least as much charm as an Elvis impersonator.

If you happen to live close to Detroit, you can see one at Marvin's Marvellous Mechanical Museum. Go to **www.conceptshed.com/projects/autowed-wedding-machine** if you're ready to commit.

DANCE DANCE DARWIN

In case you've been under a rock, Dance Dance Revolution (a.k.a., DDR), is a video game that involves "dancing" by standing on various combinations of floor sensors as instructed by a video screen in time to music of dubious quality but emphatic volume. The primary appeal of DDR, seems to be watching your friends degenerate into crazy people while playing the game. (Unfortunately, robots can't really offer this same level of entertainment.) So, what's the big deal for a robot to be tasked with playing DDR — you see an arrow, you make the movement.

Well, for the Purdue University Darwin-OP, it's not a big deal at all. A student there decided that his summer robotics research project would be to teach Darwin to play DDR. At the moment, Darwin relies on a balancing bar for stability and to enable faster moves (much the same as the safety bar on the arcade machines that humans use for essentially the same purpose). Currently in the works is tuning the robot's vision system to allow it to play DDR for real; bar-free stability may come after that.

DF RobotShop Rover

Compatible with

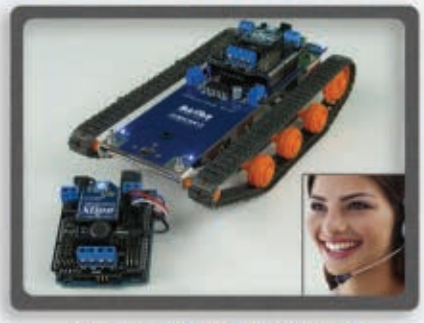
ARDUINO

MYROBOTS.COM
Connect your Robots. Reap the benefits™

Affordable, Flexible and Scalable

Compatible with thousands of off-the-shelf parts!

Wireless Control



Speech Control*

Mecanum

RobotShop
Robotics at your service!™
www.robotshop.com

Scan on your smart-phone with a bar reader app

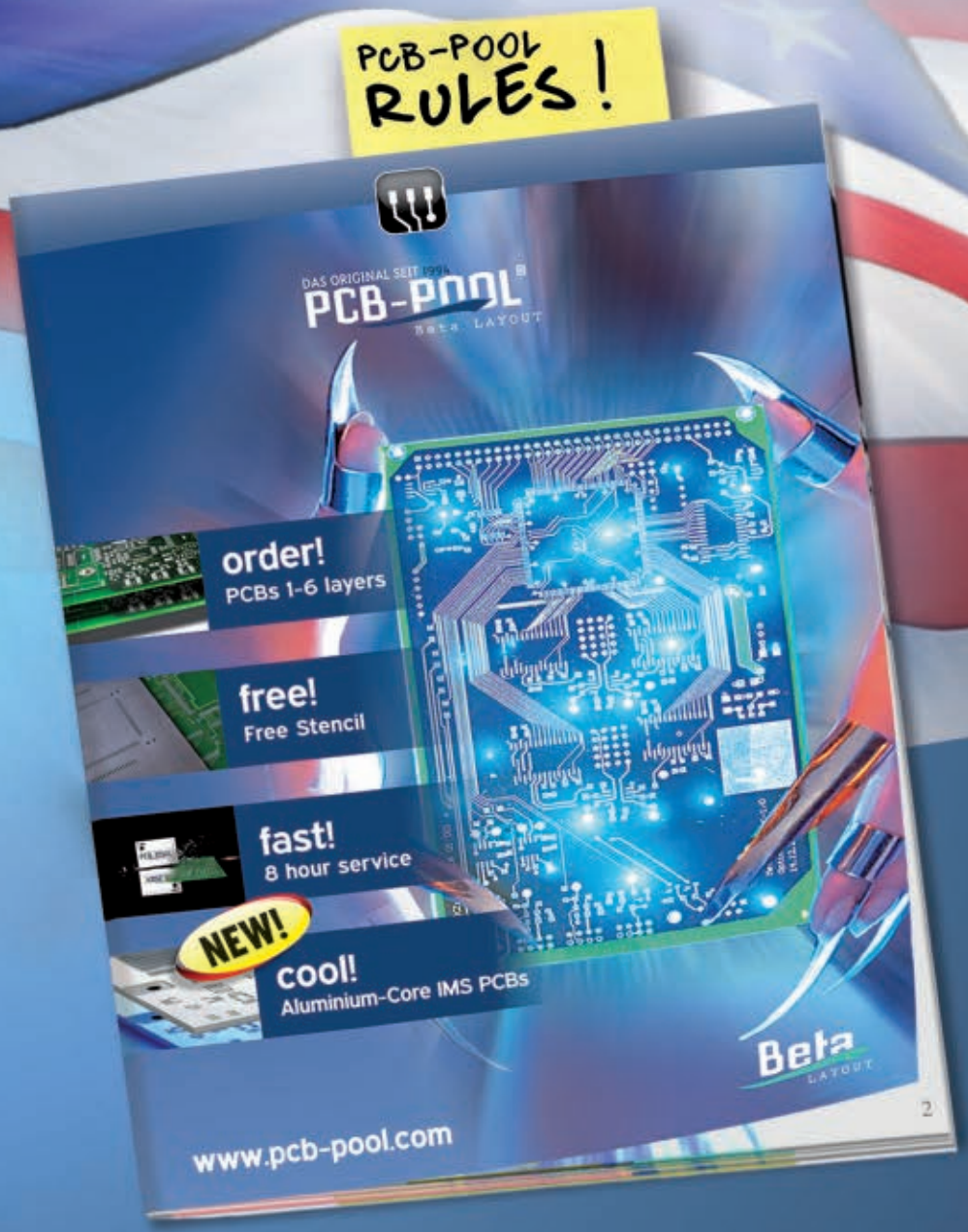
ADDITIONAL VARIANTS TO COME
THE EVOLUTION CONTINUES...

RobotShop and MyRobots logos copyright RobotShop Distribution inc. Robotics at your service and Connect your Robots. Reap the Benefits are trademarks of RobotShop Distribution inc. DFRobot logo copyright DFrobot. Arduino logo copyright Arduino. Xbee logo copyright Digi International Inc. *Headset and female operator not included with speech controlled model, which is provided with an integrated electret microphone.

THE ORIGINAL SINCE 1994
PCB-POOL®
Beta LAYOUT

Europe's leading prototype supplier now also in America

Order direct from our US facility!



email : sales@beta-layout.us
Toll Free USA : 888-977-7443

All registered brands remain the registered trademarks of the respective manufacturer !

p-cad 2006 · TARGET 3001! · Altium Designer · EDWin · cadence · GraphiCode · PROTEUS · NATIONAL INSTRUMENTS · RS-274-X · Easy-PC · Sprint Layout · PULSONIX

www.pcb-pool.com

Beta LAYOUT

COMBAT ZONE

Featured This Month:

Features

Events

www.servomagazine.com/index.php?/magazine/article/september2011_CombatZone

BUILD REPORT:

The Great Outdoors: Combat Robots vs. Mother Nature

● by James Baker

Take it Outside

Robots fight. We all know this, and it is why we read this section of the magazine. However, where, and when they fight is decided by event organizers. For many reasons — including commercial considerations — most events are held indoors, but sometimes — if we have access to small arenas — we do ask our robots to "step outside" when we want to fight.

Environmental Impact

Very few people give any thought to the effect the weather can have on their robot during a fight or demonstration. Temperature is just one variable, but it is a very important one. Let

My wedge's soft tires got too hot in the sun, so I taped them up.

us consider pneumatics for a moment. Liquid CO_2 powered weapons are reliable and powerful at room temperature, but if we take them outside we have to be more careful, especially if it is a hot day. Assuming we arrive at an event with a full gas bottle, we would empty it in our first fight, and it would develop a thin layer of ice on the outside of it as it cools. This can be a useful phenomenon if we need to keep drinks cold, but it leads to our first problem.

The frozen bottle will now accept more than its normal volume of liquid CO_2. Some people may take advantage of this in the hope of getting a few extra weapon shots out of it, but the danger comes as the sun heats the bottle up, waiting for the next fight. Overfilled bottles plus high temperatures can go one of two ways: bad or catastrophically bad, with parts of the robot failing, pressure relief valves popping, or (on at least one occasion that I know of) the bottle bursting. It is also a pain if the weather is too cold because CO_2 does not give anywhere near the same level of power or speed when the temperature is low.

Temperature affects more than just gas, as speed controllers, batteries, and motors are more prone to overheating on a hot summer day, or may suffer with power loss and short endurance on cold days. Your mega push-bot may disappoint you with poor performance and short battery life in the cold, but also in the heat as the tires and arena floor paint get extra sticky as the sun bakes them, and the robot struggles to overcome the increased friction. Remember too that batteries are very temperature sensitive in discharge, but also when they're recharging. So, watch them carefully as they may fail to charge or may burst into flames (I'm not scaring you, am I?).

The cold and heat can also affect the most basic part of the robot ... you! Taking out screws and reconnecting tiny wires is a lot harder with sweaty hands or while shivering uncontrollably, so make the servicing procedures as easy as possible. For example, by changing the type of bolts you use, you can swap your screwdriver for a hex key. This way it's harder to slip and stab yourself. Set up your serving station in the shade or under a cover, so that both you and the robot have a chance to cool down between fights. Keep a cold/hot drink in front of the charger to remind you to re-hydrate, whatever the weather.


Stand back — robot coming over the wall!

Bring the Rain

Robots are full of motors, batteries, and electronics, so there is electricity inside. This means that the robots can explode if you get them wet, correct? So, we had better run for our lives if it rains during an outdoor event then, right? Absolutely not!

Now, before I say something that can be misunderstood, it is *never* a good idea to mix electricity and water. However, it is perfectly fine to run low voltage DC batteries in the rain. Electronics are normally housed in boxes that are at least splash proof, and there are far more damaging things (such as swarf and debris) entering an open motor case, so we often cover the air vents with tape, anyway. The rain does not particularly affect radio signals over such a short distance, and very few people make their robots out of sugar, so they won't dissolve.

There are many advantages to fighting robots in the rain. For example, the universal reduction in grip allows the robots to slide and drift spectacularly, but still maintain the advantage of high grip robots as the coefficient of friction between the arena and the tires has changed, but all other factors remain the same. It also offers some nice effects as the robots splash around and create a contained mist, which on sunny days can highlight airflow patterns and even generate miniature rainbows inside the arena.


Edgehog looked great in the sunshine, but became very hot to touch.

The spiderbot stood watch as the people began arriving.

Batteries will last longer as the friction is lower, heat soak inside the robots is reduced, and high speed camera footage becomes simply spectacular, so I would highly recommend taking some simple insulation precautions to keep out the majority of the water. So, enjoy outdoor events, even if the heavens open and the rain comes down.

General Hazards

The outdoor event has many advantages, but also many common problems. If you are fighting robots outdoors, you may be doing so as part of another event — maybe a summer fair. There are likely to be other attractions, and as a result things like radio microphones, remote transmitters, special effects, and security radios flood the airwaves with their own frequencies. If you are using older radio gear, this can be a problem, since electromagnetic interference is not a nice thing to be suffering with in the middle of a field full of people and several combat robots.

Assuming you do not have radio interference, the next problem you may face is crowd control. Be sure to keep things out of reach or tethered down. Be aware also that the nice wet arena floor that is making the fights so much fun will turn you into the half-time clown if you jump in to un-stick the robots, and skate over the wall onto your face.

Practical Matters

Indoor events have everything we need, but outdoors we have certain logistical issues that are often forgotten about. Having spent mega-bucks on the latest battery charging equipment, it can be rather embarrassing to realize after the first fight that most fields do not have power outlets every few feet. If the outdoor event organizers have provided power outlets, it can be assumed that you will have only one or two at the most, and they will inevitably be fused to 3A and then blow when you plug six chargers in at once. Generators are a handy thing to have because they make you self-sufficient — assuming, of course, that the organizers have covered them in their risk assessments and allow you to run them.

Fire extinguishers are usually prevalent indoors, but not so common outside. Make sure you have suitable and sufficient fire fighting capability, especially on hot days when the grass is dry. Safely positioned trays and boxes are also useful for putting down screws and components, as small things getting knocked off the tables do not land on a nice flat floor, but disappear into the grassy carpet.

Another unexpected problem can be that of image rights. Indoor events usually state in the terms and conditions of attendance that the visitor agrees to their image appearing in photographs and video images of the event. The general public walking around their local fair have not accepted any such contract, and so any pictures or footage of the event may need their signature before it appears on the Internet or in magazines (which is why the photos in this article were taken very early in the day before the public were allowed in). These issues are simple, but almost always overlooked.

After the Event

How many of you get home from an event and dump the contents of your vehicle into the shed/garage/workshop, slamming the doors in the knowledge that the contents will not see the light of day again, until the day before the next event? If you do this after an outdoor event, you are in for a real treat when you dig out your robots for their next fight.

Robots rust even the titanium ones — in one form or another. Surface corrosion, oxidation, discoloration ... call it what you want. The humidity from the event will find every single corrodible component and rot it.

Cosmetically, this makes your robot look like rubbish, but it also has hidden implications, as electrical contacts become unreliable, bearings seize up, and motors degrade internally. It is important to look after your robot after an outdoor event.

Place it somewhere dry, re-touch any damaged or sun-bleached paint, wipe it down with water deterrent or light oil, and brush out the grass/mud/dust before they become a permanent feature. Cycle charge the batteries, check them for damage or foreign objects (grass gets everywhere!), and check and clean the motors, speed controllers, and especially the pneumatic systems. Write down any of the things that made your day difficult, or made it work well, as you will forget.

Outdoor events are fantastic when they go well. They are low cost, great fun, and low on stress. So, follow this guide, find a local fair (and a friendly small arena owner if you don't have one), and sign yourself and your robots up. If you'd like more helpful tips, go to **www.xbotz.com.** SV

PARTS IS PARTS: Gear Terminology Meshes With Little Susie

● by Morgan Berry

Every builder — even the most seasoned veterans — have experienced more than one "Oh no, what have I gotten myself into?" moment when working on a bot. Apart from the complications that can happen while building, ordering parts brings up a whole other set of issues. Even a simple wedge bot with no moving weapons can require dozens of parts, and with the dozens of varieties of those dozens of parts, an inexperienced builder might be tempted to just give up and walk away entirely. However ... fear no more! The Combat Zone's guide to gears is sure to make one of the most intimidating steps in the build process much simpler.

Meet little Susie. After attending a combat robotics competition, little Susie was inspired to build a metal-tearing, pain-bringing death machine of her own. A trip to the local hobby shop left Susie with a bargain box with a variety of parts. Now, all she needs to build her bot is a shell and gears.

The metal shop teacher at her school agreed to help her build a shell. So, smiling at how simple the process is, Susie sits down at the computer to purchase some gears. Susie's motors run at 1,000 RPM, and she is envisioning a wedge bot, so she needs to increase her torque to provide the pushing power a wedge needs. Susie calculates that her bot's wheels should turn at 100 RPM. She'll reduce the RPM using gears. (If she was using her motor to drive a saw blade for a weapon, Susie would instead need to increase the RPM to enhance the cutting ability using the same gear ratio.)

FIGURE 1

After clicking onto the first website, the smile quickly fades from Susie's face. Pressure angle, bore size, diametral pitch? Susie is completely lost. She realizes she has a lot to learn about gears.

The first thing Susie researches is the type of gear she needs. She quickly rules out a rack and pinion gear (**Figure 1**) which is used to convert circular rotation into a linear motion, or a bevel gear (**Figure 2**) which is used to change the direction of the gear rotation, usually by 90°. Since her wheels turn in the same direction her motor spins, she doesn't need to worry about direction of rotation (but, if Susie's bot had a blade mounted on top, a bevel gear might be necessary). She also rules out a worm gear (**Figure 3**) which is used when large reductions in RPM are needed.

Now she must decide between the two remaining types of gear: a spur gear or a helical gear. A spur gear (**Figure 4**) is the most common gear and has parallel teeth, so when they rotate, each set of teeth on the gears come into direct contact with

FIGURE 2

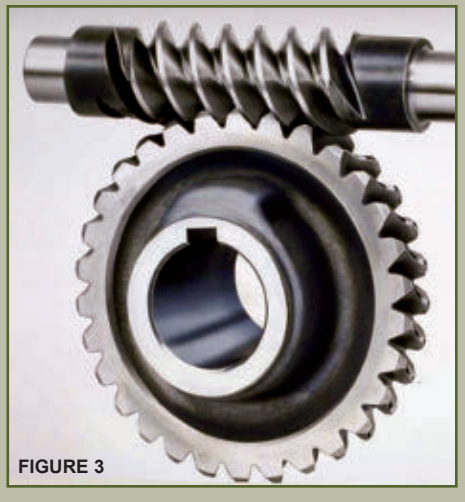


FIGURE 3

SUSIE'S LIST OF ADVANCED GEAR TERMS

While you probably won't need to worry about many of these specifications when deciding what gear to purchase for your bot, it is important to understand the following terms in order to have a working knowledge of how a gear works. Because gears are circular, you should also be familiar with circle terminology (diameter, chord, tangent, etc.).

- *Addendum:* The distance from the pitch circle to the addendum circle.
- *Dedendum:* The distance from the pitch circle to the root circle.
- *Outer Diameter:* The diameter of a gear measured from the top of the teeth.
- *Root Diameter:* The diameter of a gear measured from the base of the teeth.
- *Pitch Diameter:* The diameter of a gear measured from the pitch circle.
- *Addendum Circle:* An imaginary circle drawn at the top of the teeth.
- *Root Circle:* An imaginary circle drawn at the base of the teeth.
- *Pitch Circle:* An imaginary circle drawn at the point of contact when two gears mesh together.
- *Circular Thickness:* The thickness of a tooth measured as an arc along the pitch circle.
- *Chordal Thickness:* The thickness of a tooth measured between two chords that extend from the points where the pitch circle passes through both ends of a tooth. This measurement will be slightly less than the circular thickness, as it is a measurement taken from a straight line whereas circular thickness is the measure of a curved line.
- *Circular Pitch:* The distance between a point on one tooth and the corresponding point on the next tooth; the distance between two teeth. To function properly, two corresponding gears must have the same pitch.
- *Whole Depth:* The measurement of the distance between the top of a tooth and the bottom of a space between gears.
- *Working Depth:* The amount of space the tooth of one gear occupies when meshing into the space between two teeth on the corresponding gear.
- *Clearance:* The amount of space between the top of one gear and the bottom of the space between two teeth on the corresponding gear. When two gears mesh, the tooth of one gear does not actually come into contact with the base of the second gear. Simply put, this is the gap between the tooth of one gear, and a space on another gear.

Toolbox Tearout

Cut out this sheet or download the file at the article link and keep it handy as a quick reference for the next time you sit down to plan a bot. These terms are practical and will influence how well your gears work.

each other. The helical gear (**Figure 5**) has angled teeth to reduce the impact between two teeth while they rotate. Helical gears are used when noise reduction is important (less impact = less noise), but since Susie doesn't mind a loud bot, she decides a spur gear will do the job.

Now that she has determined the type of gear, Susie turns her attention to the issue of how to reduce her RPM. She discovers that the key is the gear ratio. A gear ratio is the relationship between the number of teeth of two gears. Susie can figure out the gear ratio she needs by working backwards from the existing RPM of the motor and the desired RPM of her wheels. Remember that she wants to reduce her RPM from 1,000 to 100, so as a ratio this would be 1,000:100 or 10:1.

Keeping this ratio in mind, Susie looks through the spur gears on the online store and finds one with 12 teeth and one with 120. This will give her the increased torque she needs for her bot. When she is building her bot, she will attach the smaller gear to her motor and the larger one to the wheel so that after 10 spins of smaller gear, the larger gear will spin once, thus slowing down the wheel's RPM.

The gears Susie selected also have options of "with" or "without" hubs, face width, and material. Hubs are the protruding part of a gear that have a hole for attaching to a motor or wheel using a setscrew. This is a must for Susie's bot. As for the face width (the width of the gear teeth), it is important to keep in mind the purpose of the gear.

Newton's third law says that for every action, there is an equal and opposite reaction. In the world of combat robotics — where the bot is being designed for collisions — the gear teeth will be exposed to a lot of reactive force. If the teeth are exposed to too much force, they could become stripped. So to compensate for this, it is generally better to use a gear

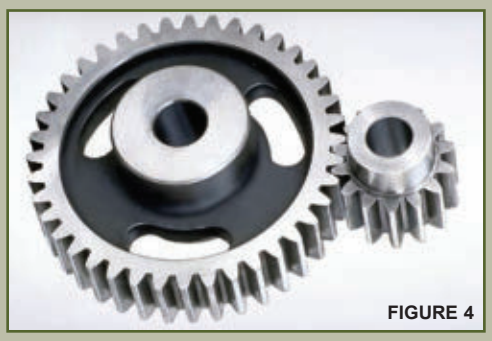


FIGURE 4

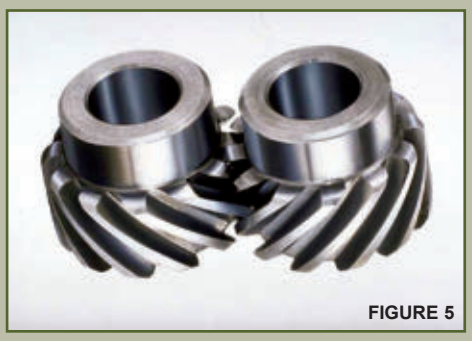


FIGURE 5

- *Gear Type:* Use a rack and pinion gear to change circular motion to linear motion. Use a bevel gear to change the direction of rotation. For a large reduction in RPM, use a worm gear. For everything else, use a spur or a helical gear.
- *Gear Ratio:* The relationship between the number of teeth on two gears. Changing the gear ratio will influence the amount of torque or speed a gear has. To increase torque, a smaller gear should be mounted on the motor and a larger one on the wheel. To increase speed, do the opposite.
- *Hubs:* The protruding part of the gear that contains a bore for mounting gears. Using a gear with a hub is generally recommended because if you use a hubless gear, you will need to bond the gear to the shaft by using glue, epoxy, or solder. This also makes replacing the gear much more difficult, so if you do use a hubless gear, use a more durable metal one in lieu of plastic.

Hub Diameter:

- *Diametral Pitch:* The number of teeth in an inch of the pitch diameter. The diametral pitch of two corresponding gears determines if the gears will mesh together properly. The ratio of the number of teeth between two gears must be the same as the ratio between the two gears' diametral pitch.
- *Bore Size:* The diameter of the bore (hole used for mounting gears). It is obviously important to choose a bore size that will allow you to mount the gear properly. The bore size should be slightly bigger than the motor shaft on which you are mounting it and should fit snuggly. If you find yourself unsure about which size to choose, it is better to choose a slightly smaller size which can then be reamed out with a drill to the size you desire.
- *Pressure Angle (angle of obliquity):* The pressure angle is a term that shows the angle between the line of contact (an imaginary line drawn through the point where gears come into contact from the beginning to the end of meshing) and the tangent line (an imaginary line drawn at the point of contact between the two pitch circles of corresponding gears). For practical purposes, it is really only important to note that the pressure angles of two corresponding gears must be the same, and that the smaller pressure angles tend to indicate weaker gear teeth. Older gears typically have a 14.5° pressure angle; newer gears most often have a pressure angle of 20° or 25°.
- *Face Width:* The width of the gear teeth. Because combat robots are exposed to a lot of force, using a gear with a wider face width can distribute the force of impact during a fight over a wider surface area, making it less susceptible to damage.
- *Material:* Gears come in plastic and a variety of metals such as aluminum, brass, or steel. Plastic is obviously cheaper and lighter, but more susceptible to wear, while metal can add weight and expense to your bot but are more durable.

with a larger face width because the increased surface area will distribute the force over a larger area.

Susie selects the highest face width that will fit inside her Ant-weight shell. For the material, Susie chooses plastic because it is less expensive and weighs less than other materials. If Susie was building a bigger bot, she might want to consider a metal gear that could better withstand the force of an impact with another heavy bot.

After her research, Susie has learned enough about gears to make her basic bot, but she has barely scratched the surface of gear terminology. She resolves to learn more about gears and eventually has an entire list of advanced nomenclature, which she kindly provided with this article. SV

Figure 1 is courtesy of ***www.directindustry.com****. Figures 2-5 are courtesy of* ***www.howstuffworks.com***

EVENTS

Completed and Upcoming Events

Completed Events for June 2011

Gulf Coast Robot Sports-7 was presented by Gulf Coast Robot Sports in Bradenton, FL on June 11th.

Upcoming Events for Sep–Oct 2011

House Of Robotic Destruction 2011 will be presented by the Ohio Robot Club at Classic RC Raceways in Akron, OH on September 17th. Go to **www.ohiorobotclub.com** for further information.

Mecha-Mayhem 2011 will be presented by the Chicago

Robotic Combat Association in Rosemont, IL, on October 22nd and 23rd. Go to **www.thecrca.org** for further information.

Robot Battles 41 will be presented by Robot Battles at DragonCon in Atlanta, GA on September 4th and 5th. Go to **www.robotbattles.com** for further information.

ROBOT BATTLES™

ComBots Cup VI will be presented by ComBots in San Mateo, CA on October 29th and 30th. Go to **http://robogames.net/registration/event/view/11** for further information. SV

EVENT REPORT: 2011 NRL National Championships

● by Nicholas Ozorak

It was may 21, 2011. In an airplane hanger at the edge of the Indianapolis International Airport, I could hear crashes, clanks, and the hum of tools being used. However, this was not related to airplanes — this was the 2011 National Robotics League National Championships.

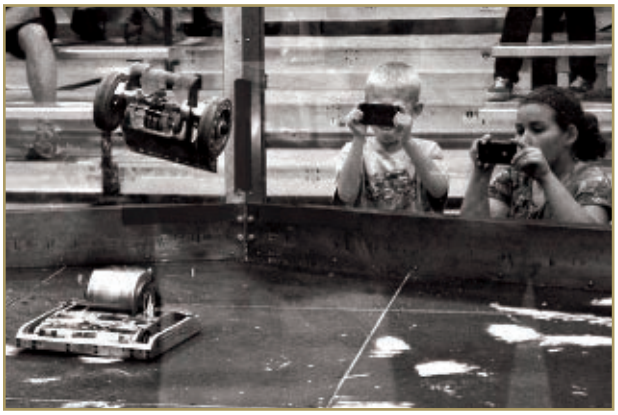
The venue was the Vincennes University Aviation Technology Center, with a full-size Boeing 737 for company. Underneath the plane's left wing was the pit area, where 29 teams would make last-minute adjustments and prepare their robots for battle.

Once the teams arrived at the venue, they unpacked their tools and set up shop. Their first challenge was to make sure their robot passed inspection to ensure that it was both legal and relatively safe to fight in the arena. Robots that didn't make the weight limit or needed to be recalibrated were sent back to the pit for fine-tuning.

The championship was structured as a double-elimination tournament. For three minutes per round, each opponent's goal was to disable the other's robot. The first loss meant being knocked into the loser's bracket, and the second loss meant the end of the line for that robot.

Occasionally, the robots that

This article reflects the views of the author, not necessarily those of the National Robotics League or the National Tooling and Machining Association. Thanks go to Elizabeth Weiss Ozorak for comments on an earlier draft of the article.

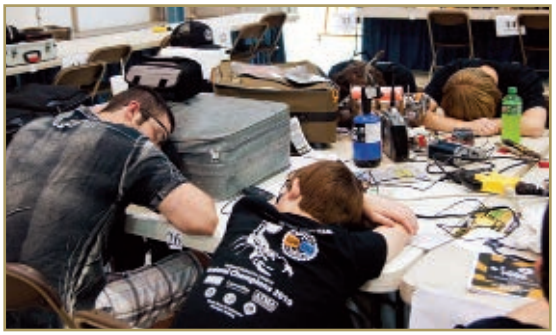


had lost would engage in a rumble to see how many could fight in the arena at once, and who would prevail. Audience applause was used to judge the rumbles.

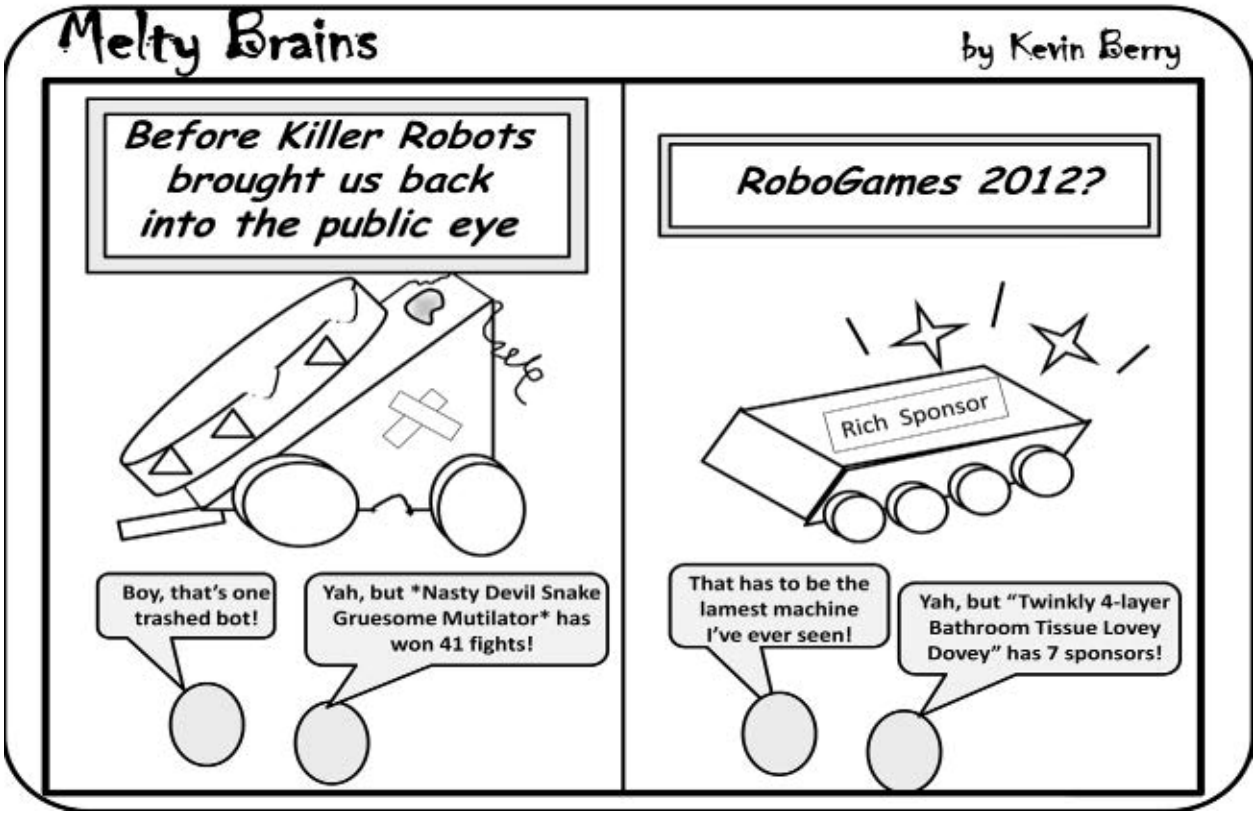
Over the course of the weekend, 56 one-on-one robot fights took place in the decagonal arena. By Sunday morning, just a few robots remained in contention. The final matches were exciting, punctuated by showers of sparks and flying metal parts. Robots were tossed into the air and bounced off the arena walls.

The Champion's Trophy went to Bloomsburg Area High School's Pixie, based on a combination of effective craftsmanship, documentation, and driving.

Having helped to build and drive a robot myself in the past, I was pleased to see a new generation hard at work on unique machines. It's clear that this program immerses kids effectively in robotics and manufacturing.



It's also about the pleasures and challenges of teamwork and sportsmanship ... and the thrill of watching the robots strike their opponents into submission. **SV**



Going Further With The Beginner Bot

Part 2

Yesterday's robots can teach us a lot about today's robots. That's because old school bots used simpler ways to control them. It was easier to understand what was happening, thanks to their reliance on simple circuits. You could see how the wires connected from one point to the next, and visualize how the juice from a battery turned on this component, or switched off another.

by Gordon McComb

www.servomagazine.com/index.php?/magazine/article/september2011_McComb

There's no need to stock up on tubes and giant relays ... we don't need to go *that* far back in history! If you'd like to step into the world of smart robotics, however, a great way is to learn how to build a no-frills, no-brain mechanical pet that uses basic electronic parts to move around the floor.

What's even better is the parts for your bot can be re-purposed when you're ready to move to the next step. That's the idea of the Beginner Bot, introduced last month. In Part 1, we discussed how to construct the Beginner Bot using common materials like plywood or sheet plastic. We also discovered how to drive and steer our bot by using a pair of mechanical switches.

This time, we'll go to the next level. We'll replace the manual switch control with fully automatic — yet simple — electronic function. Our robot will react to light, coming toward any bright light source.

Using just a flashlight, you'll be able to lead your robot around a darkened room. (Yeah, I know I was going to demonstrate tactile feedback this time around, but I'm going to save that for a future installment.) **Figure 1** shows the completed Phase 2 Beginner Bot — all ready to be guided by nothing more than photons.

Remember that the techniques you'll learn here — and even the components you'll use — will be applied in upcoming episodes of the Beginner Bot series.

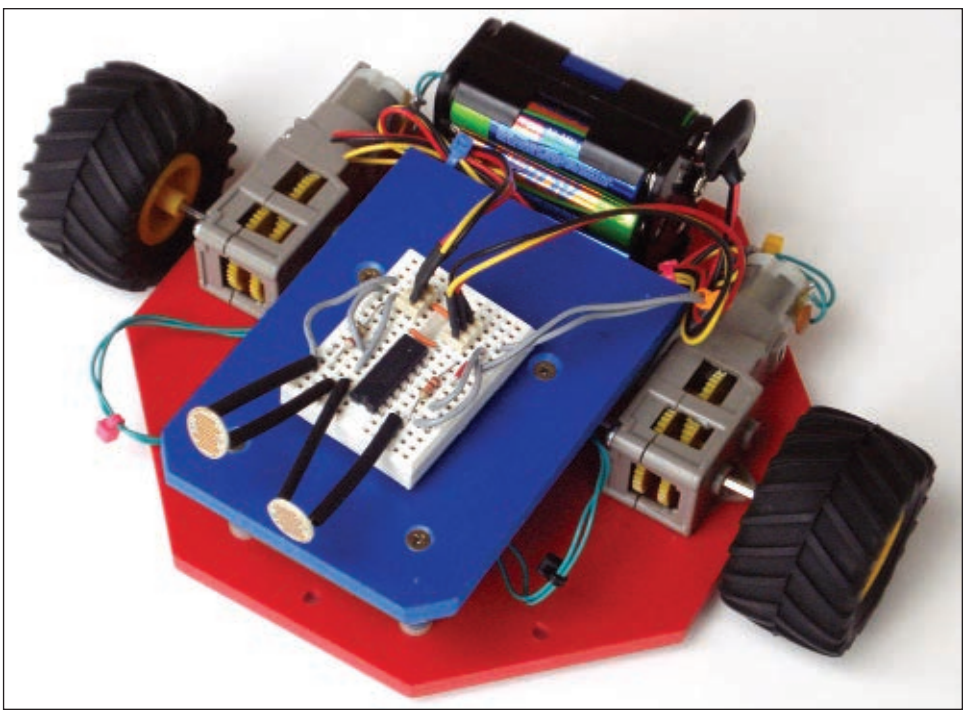


FIGURE 1. Completed Phase 2 Beginner Bot with snail-like "eyes" for following bright lights.

Using Electronic Motor Control

Mechanical switches help demonstrate how to control the motors of a robot. As you discovered in Part 1, motor direction is controlled by alternating the polarity of the current applied to the motor's terminals. A double-pole, double-throw (DPDT) switch is a great demonstrator of robot motor control.

A manually controlled robot is nothing more than a fancy toy; it's not a true robot until it can run on its own. Through electronic control, you can replace the switches with

H-Bridge Specifications

Phase 2 of Beginner Bot calls for a Seeed Studio L298 dual H-bridge motor control module, available online from **RobotShop.com** and other retailers. This module provides the L298 integrated circuit on a heatsink, plus the required protection diodes, operating LEDs, and terminal blocks for wiring to a power source and motors.

You can use most any other L298 module, or even another module using a different type of H-bridge, as long as it conforms to these basic specifications:

- Capable of driving up to two amps per motor.
- Three inputs for each motor: enable, and two differential inputs. (Avoid the type with just two inputs for direction and on/off; although these will work, it will require you to revise the wiring from the 74HC14).
- On-board five volt voltage regulator.

You can make your own H-bridge using an L298 chip. Be sure to check the datasheet for the device, and add the required protection diodes on each of the motor terminal outputs.

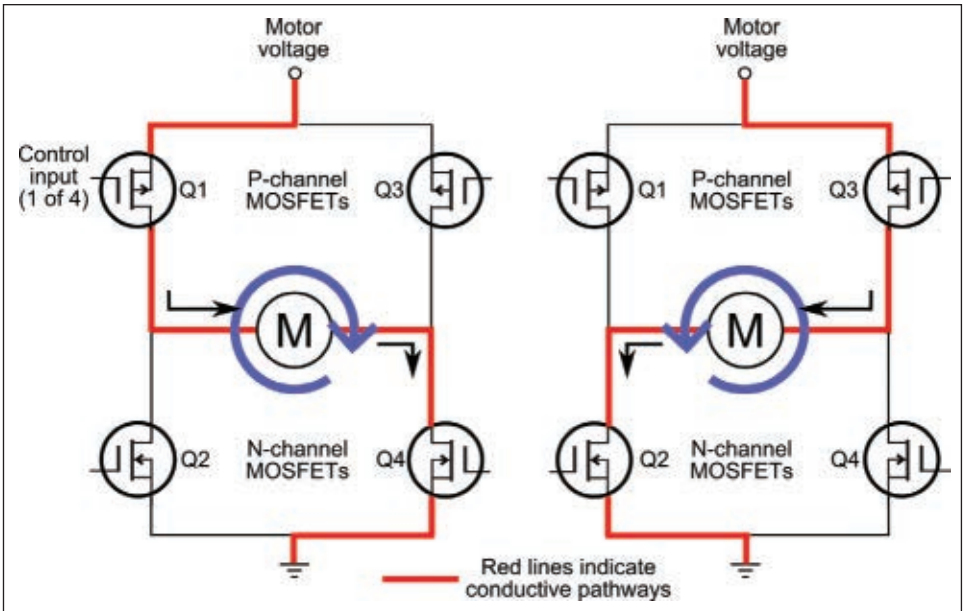


FIGURE 2. Concept of the electronic H-bridge. The direction of current flow determines which way the motor turns.

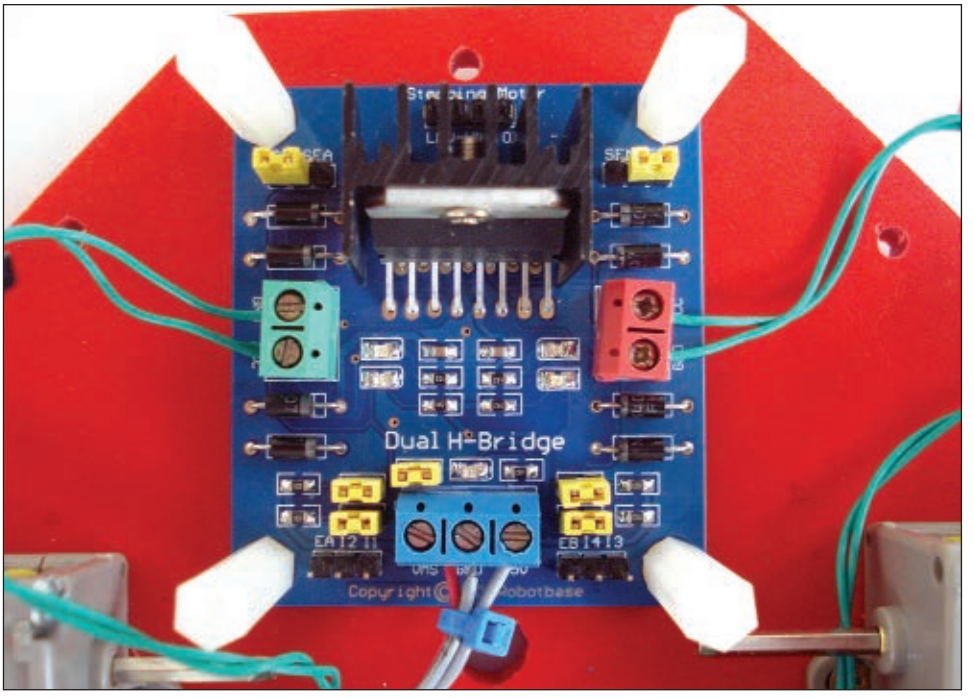


FIGURE 3. Ready-made H-bridge module. This one is from Seeed Studio, but a similar model will work too.

circuitry. This circuitry effectively duplicates the action of those DPDT switches. And because the circuitry can be operated electrically, other electronic components can be connected as sensors to provide automatic function.

An H-bridge circuit is the most common way to provide all electronic control of a motor. It's called an H-bridge because the schematic diagram of the circuit typically shows the main components — usually transistors of one type or another — in an "H" pattern. **Figure 2** shows a simplified H-bridge (don't try to build this circuit; it's just for demonstration).

While there are many — and I mean *many* — workable H-bridge designs, most of the ones you can easily replicate leave something to be desired. Most rely on big and bulky transistors, so the finished circuit is ungainly. To make the thing easier to construct at home, some components might be omitted which could affect its efficiency.

It's often easier (and sometimes cheaper!) to get a ready-made H-bridge. A popular one is the L298 — a complete H-bridge in a single integrated circuit. One L298 will operate two motors. The L298 comes in a special kind of high wattage package that isn't designed for breadboarding, and it needs some external parts — specifically, a set of protection diodes — to complete its circuitry.

For the sake of convenience, I like to use the many ready-made L298 circuit board modules available. Most are priced between $15 to $35, depending on features. The one I've specified in this article is from Seeed Studio (see **Figure 3**), and retails from RobotShop and other online shops for under $25. You're free to use another H-bridge module, as long as it conforms to the specifications in the **sidebar** found elsewhere in this article.

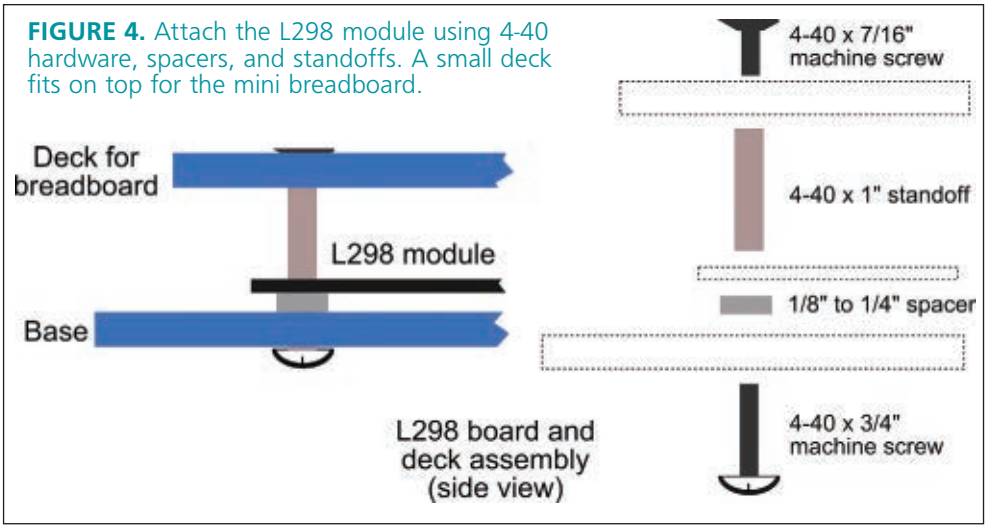


FIGURE 4. Attach the L298 module using 4-40 hardware, spacers, and standoffs. A small deck fits on top for the mini breadboard.

Mounting the L298 Module Board

The first order of business is to remove the switch panel and wiring from the Phase 1 Beginner Bot. These components won't be used this time around, but don't discard or recycle them just yet! You'll have the option to reuse these components in later installments of this series.

Note: Rather than unsoldering the wires to the motors, completely remove the motors from their gearbox housing. Merely pull back on the retainer clip on the side of the gearbox, and lift out the motor. Set the motor/switch assembly aside for the moment.

Drill four holes into the Beginner Bot base for mounting the L298 module board. Be sure to position the board on the end *opposite* the skid. The reason: You want the heaviest components over the skid to keep the robot from tipping the other way. You'll be placing the batteries — which weigh more — on the skid end.

Orient the L298 module so that the black metal heatsink faces toward the far end of the robot. We'll call this the "front" of the Beginner Bot.

Use 4-40 x 3/4" machine screws, spacers, and threaded standoffs to mount the board as shown in **Figure 4**. The Seeed Studio L298 module leaves plenty of room for mounting the board, but not all H-bridge boards are so cooperative. If mounting space is tight, you may need to use smaller 2-56 size hardware which is harder to find.

Also shown in **Figure 4** is how to mount a separate small deck for a mini breadboard. See **Figure 5** for a cutting and drilling template for this deck. Use 1/4" plywood or expanded PVC plastic. Don't attach this deck quite yet, because you need to first wire up the L298 module.

Replacing the Tamiya Gearbox Motors

The Beginner Bot uses a pair of low cost gearbox motor kits from Tamiya. The small DC motor that's included in the kit is rated for operation at just three volts. As motors go, it's not terribly efficient. As a result, these motors consume a lot of current, especially if they get bogged down under lots of weight or strain.

All H-bridge circuits are rated to handle a certain amount of current before they overheat — and either shut off, fail, or become damaged. The L298 module can handle up to two amps per motor. Yet, the Tamiya motors can consume in excess of three or four amps when powered at just four to six volts.

Fortunately, you can get low cost drop-in replacements for the stock motors provided in the Tamiya gearbox kits. For example, the Pololu #1117 replacement motor is rated for operation at six volts (can be operated at up to 12 volts), and at that voltage consumes a maximum of only 800 mA (0.8 amps) when stalled — the motor is still powered but cannot move.

When replacing the motor, you'll need to pull the small spur

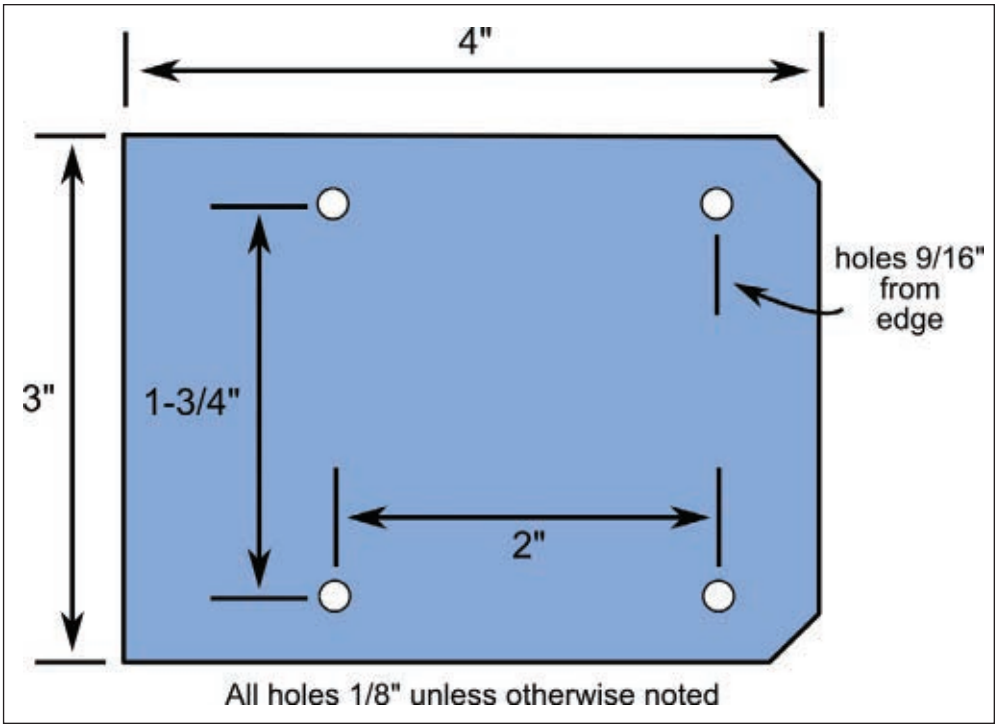


FIGURE 5. The cutting and drilling layout for the small breadboard deck. Be sure the holes in the deck match the position of the holes in the L298 module board.

gear off the old motor and mount it onto the replacement. The fit is tight; avoid grabbing the teeth of the gear with a tool to remove it. Instead, place the jaws of a pair of small needlenose pliers behind the gear, and gently work the gear off. Be careful that the gear doesn't come flying off the motor shaft or you may never find it!

Solder 7" to 8" leads (use 18 to 20 AWG stranded wire) to the motor for connecting it to the L298 module. Here's a tip: It's often a good idea to add a small 0.1 µF ceramic disc capacitor directly between the terminals of the motor along with the connecting wires. The capacitor helps to reduce any negative effects of electrical noise induced by the motor. Be sure the capacitor is rated at 25 volts or higher.

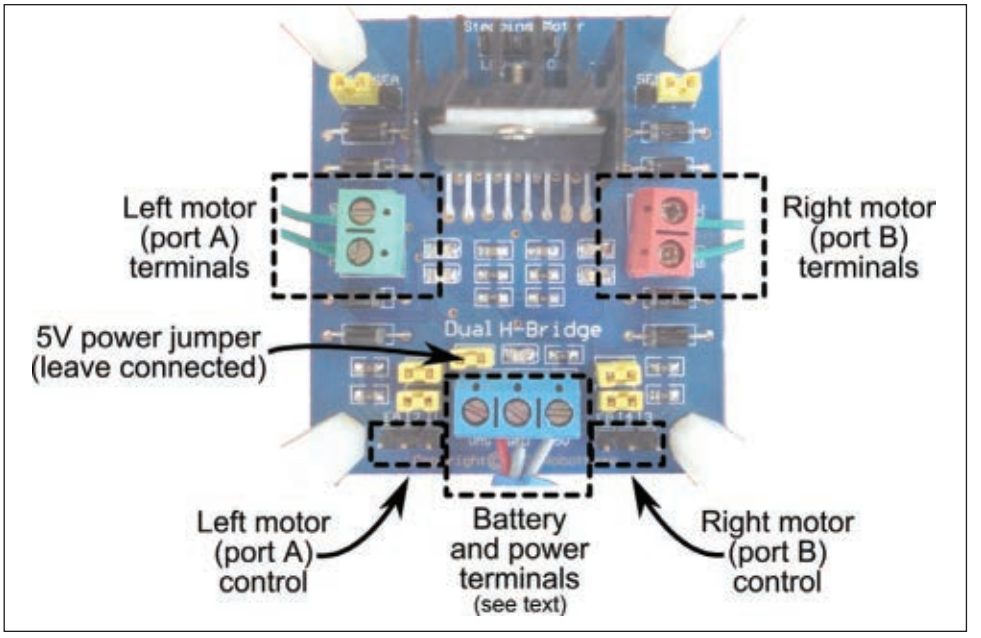


FIGURE 6. Note the main points of interest on the Seeed Studio L298 board. Check the datasheet for this product for a description of the jumpers. For the Beginner Bot, leave everything factory set.

Connecting the Motors to the L298 Board

Locate the two-terminal motor terminal blocks on either side of the L298 board (see **Figure 6** for the main points on the Seeed Studio L298 module). On each block, the terminals are labeled VCC and Gnd. Orient the robot so that the heatsink on the L298 board faces away from you. Then:

- Attach the wires from the left motor to the motor terminals on the left (shown as green in the illustrations).
- Attach the wires from the right motor to the motor terminals on the right (shown as red in the illustrations).

Use a small flat-bladed screwdriver to tighten the terminals so the wires remain snug.

Attach Battery Holder and 9V Battery Clip

The three-cell battery holder used in the Phase 1 Beginner Bot doesn't provide enough voltage for operating the L298 H-bridge. As a replacement, use a six-cell, double-sided (three cells per side) AAA or AA battery holder. Get the kind with the two tabs for connecting with a polarized nine volt battery clip.

Locate the three-terminal block at the bottom of the L298 module. The strip has connections labeled VMS, Gnd (for ground), and +5V. The +5V terminal provides five volts, supported by the built-in five volt regulator on the bottom of the L298 module.

Wire the nine volt battery clip to the L298 module. The red (+, positive) lead connects to the power terminal marked VMS. The black (–, negative) lead connects to the power terminal marked Gnd.

Prepare a 7" length of 22 AWG solid conductor wire by stripping off 1/4" from both ends. Insert one end of this wire to the same Gnd connection as the black battery lead. Prepare a second 7" length of 22 AWG solid conductor wire in the same manner as above. Insert one end of this wire into the +5V power terminal. See **Figure 7** for a view

USING BATTERY CLIP TERMINALS AS A SWITCH

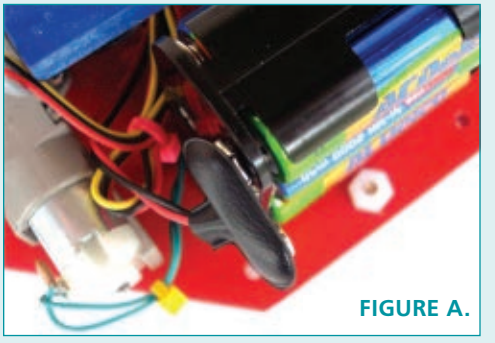


FIGURE A.

Battery holders equipped with polarized terminals for nine volt battery clips provide both an electrical connection and a cheap form of switch. Here's how: Snap on just one terminal, and turn the clip so that there's no contact to the other terminal. This is the off position. To turn your robot on, complete the battery connection by rotating the clip so the other terminal touches its mate.

For this to work, the terminals must have a tight fit. If there's too much slop, carefully squeeze together the prongs of the terminals to tighten things up. Don't go overboard, or you'll bend the metal out of shape.

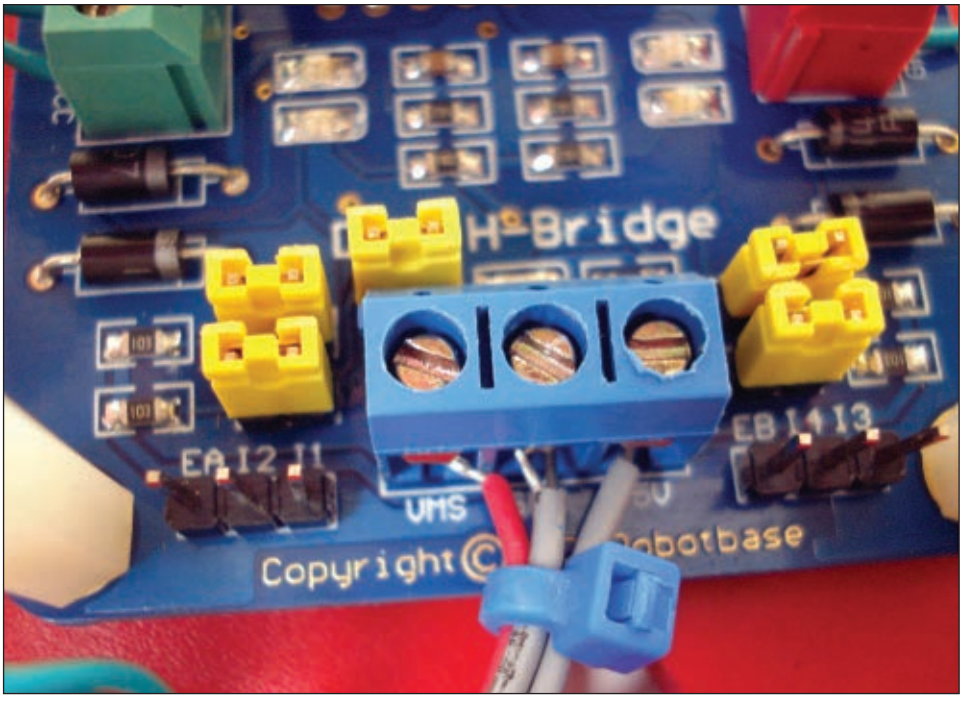


FIGURE 7. Close-up of the power terminal. The L298 board provides its own regulated five volts for the circuitry on the breadboard.

of the wired power terminal. I've used a small cable tie to keep everything together.

Use a small flat-bladed screwdriver to tighten the terminals so the wires remain snug. You DO NOT want any of these wires to accidentally come loose, or else they could cause a short circuit and damage to the L298 module.

Attach the battery holder to the Beginner Bot base using two 1" Velcro™ squares as shown in **Figure 8**. This arrangement allows you to pull the holder off the base to replace or recharge the batteries.

For the prototype Beginner Bot, I used a set of six AA size nickel-metal hydride (NiMH) cells. You can use AAA size if you wish, and substitute nickel-cadmium (NiCd) or alkaline cells. I prefer NiMH batteries because they're a little more environmentally friendly. Though they initially cost more than alkaline cells, when you factor in recharging them hundreds of times, they're a very good deal. (Be sure to use a recharger made for NiMH batteries.)

When using NiMH or NiCd batteries, the six-cell pack delivers a nominal 7.2 volts. When using alkaline batteries, the pack delivers nine volts. Either voltage is acceptable. However, be sure to recharge your NiMH or NiCd batteries when the pack voltage falls under seven volts.

Add the Solderless Breadboard Deck

Augmenting the L298 motor module is a small collection of circuitry to provide the Beginner Bot with sensitivity to light. Rather than solder together this circuitry, you can use a small solderless breadboard. A mini breadboard with 170 contact points will do.

The control circuit uses two cadmium sulfide (CdS) photocells, an inexpensive and readily available 74HC14 integrated circuit, a couple of resistors, and some wire. The solderless breadboard connects with the L298 module via the two power wires you provided earlier, plus a pair of 10" length three-conductor male-to-female R/C servo extensions.

At the heart of the control circuit is a CdS photocell, also called a light sensitive resistor. It's connected as shown in **Figure 9** with a 22 kΩ resistor to form a voltage divider. The output of a photocell is a varying resistance — the darker it is, the higher the resistance. With the resistor added and a connection point in between, the output becomes a voltage that varies between zero and five volts, depending on the brightness of the light.

The 22 kΩ resistor helps establish the sensitivity of the light sensor. The

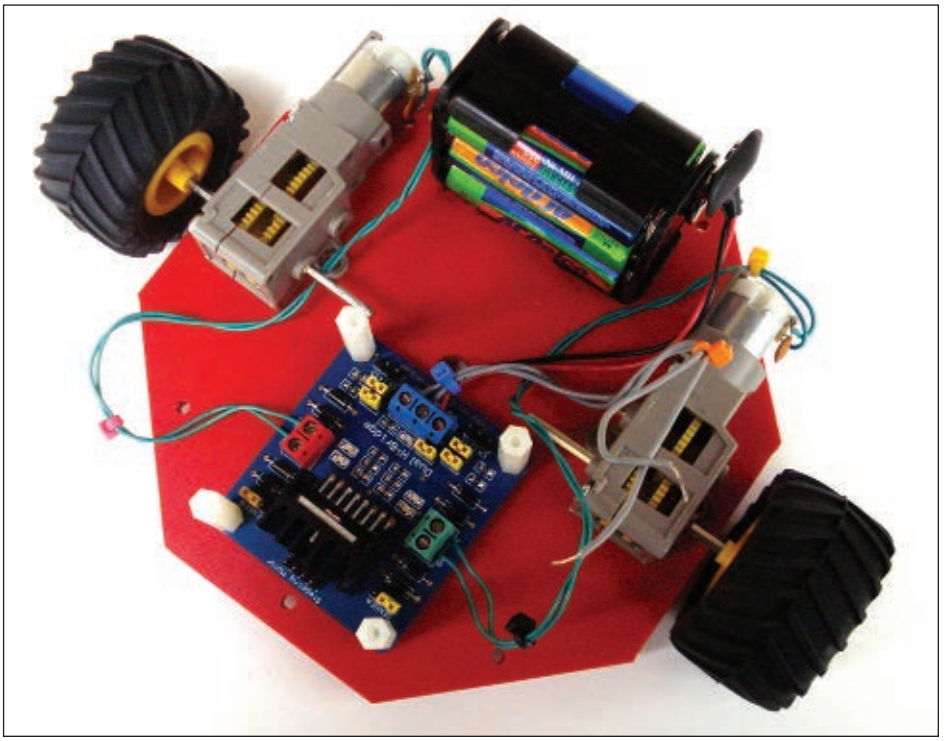


FIGURE 8. The Beginner Bot with L298 board and batteries installed, ready to go. The battery holder is held in place with Velcro.

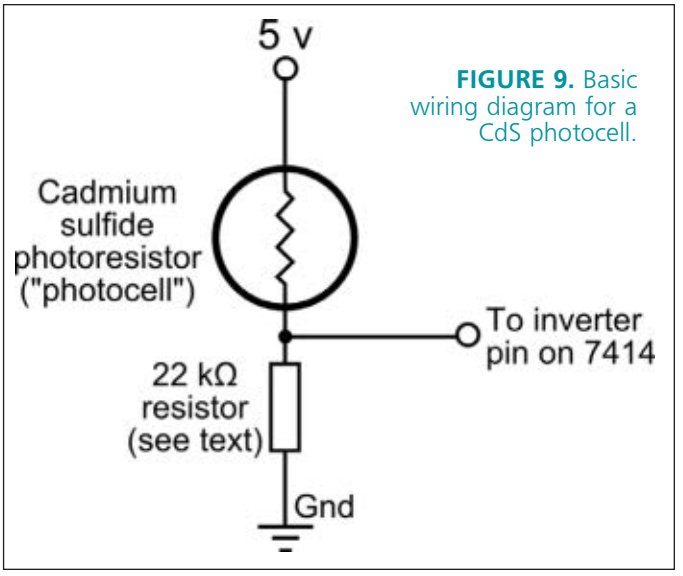


FIGURE 9. Basic wiring diagram for a CdS photocell.

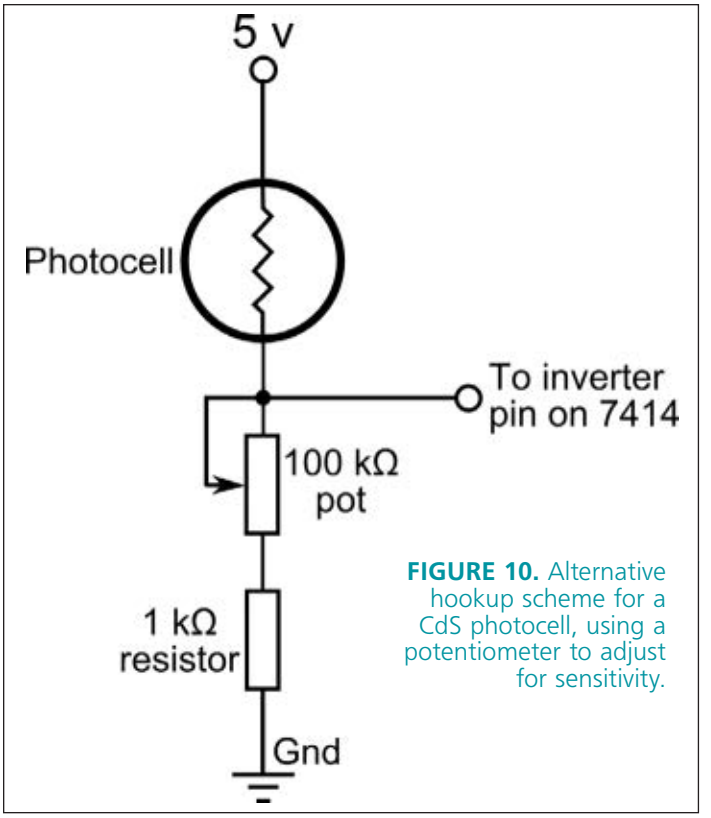


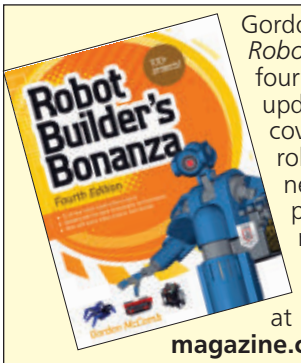
FIGURE 10. Alternative hookup scheme for a CdS photocell, using a potentiometer to adjust for sensitivity.

higher the value, the more sensitive it is — but the less linear the photocell's response to light. Different photocells exhibit different output behaviors, so you need to select a resistor value that best matches the photocell and the light conditions you expect.

I selected the 22 kΩ value because I wanted to operate the Beginner Bot in a mostly darkened room, and lead it using an LED flashlight. Use trial-and-error to find the best value for your components, or else use the circuit in **Figure 10** which adds a potentiometer for adjusting sensitivity.

Remember that you need two photocells and resistors (and potentiometer): one for the right motor, and one for the left motor. (And no, it's not a mistake that the photocell on the left controls the right motor, and vice versa. This arrangement establishes the Beginner Bot as *favoring* light; if you reverse the connections, it will behave as if it's *avoiding* light.) The output of each photocell is a voltage applied to one of six inputs of a 74HC14 Schmitt inverting buffer. This one integrated circuit contains six separate buffers; we'll be using four of them in this project. With a Schmitt buffer, the circuit triggers at pre-defined input levels, called thresholds. This "snap action" provides clean on/off switching, even though the light falling on the photocell is a varying voltage.

(The exact voltage of the switching threshold depends on the supply voltage to the chip — whether the voltage change is positive-going or negative-going — and variations that can occur from one chip to another. For a 74HC14 running at five volts, the positive-going threshold voltage is around 2.4 volts; the negative-going threshold voltage is about 1.5 volts.) In total darkness, the output of the photocell is zero volts (more or less). As the 74HC14 is an inverting buffer, it will take this zero volts, and the output will be five volts. As the light increases, the buffer does not



Gordon McComb is the author of *Robot Builder's Bonanza*, now in its fourth edition. Greatly expanded and updated, this best selling book covers the latest trends in amateur robotics, and comes with 10 all new robot construction projects, plus more ideas for building robots from found parts. Look for *Robot Builder's Bonanza, 4th Ed* in the *SERVO* Webstore at **http://store.servomagazine.com**.

Gordon may be reached at rbb@robotoid.

Sources

Precut and predrilled Beginner Bot base, with all construction hardware:

Budget Robotics
www.budgetrobotics.com

Seeed Studio L298 Dual H-bridge module, IC:

Seeed Studio
(item MOT103B1M)
www.seeedstudio.com

Robotshop
(item RB-See-103)
www.robotshop.com

Trossen Robotics
(item SEEED-MOT103B1M)
www.trossenrobotics.com

Compatible L298 H-bridge modules (selected listing only):

Hobby Engineering
www.hobbyengineering.com

Pololu
www.pololu.com

SparkFun
www.sparkfun.com

Solarbotics
www.solarbotics.com

Replacement FA-130 size high efficiency Mabuchi motors:

Pololu
item #1117
www.pololu.com

Plus, you need (available from many online and local retailers):

Six-cell AAA (preferred) or AA battery holder with nine volt battery clip terminals; nine volt battery clip; 22 gauge solid and stranded conductor hookup wire; Velcro™ squares; assorted assembly hardware as per the text; 170 contact point mini breadboard; two 22 kΩ resistors; two large CdS photocells; 74HC14 hex inverter IC; and heat shrink tubing.

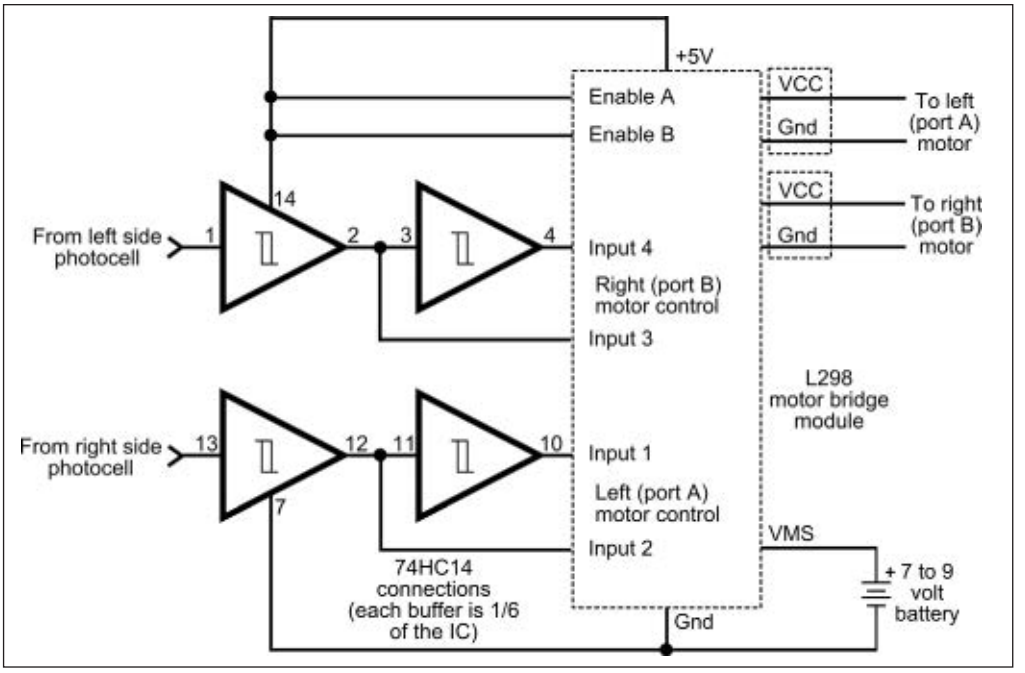


FIGURE 11. Schematic diagram for wiring the 74HC14 inverting buffer IC and other components. Connections are shown for attaching the circuit to the L298 module.

TABLE 1		
Input A	**Input B**	**What Happens**
0 volts	0 volts	Motor stops
0 volts	5 volts	Motor turns one direction
5 volts	0 volts	Motor turns the other direction
5 volts	5 volts	Motor stops

react until the voltage from the photocell reaches the threshold value. At that point, the output of the buffer immediately switches to zero volts.

Figure 11 shows how each motor uses two inverting buffers. They're wired in such a way that there are two *complimentary* (opposite) outputs: When one output is zero volts the other is five volts. This complimentary voltage state is needed for the L298 H-bridge.

Each motor has two control inputs. The motor will turn only when one input is zero volts and the other is five volts. The direction of the motor is reversed by swapping the voltages at each input.

Figure 12 shows the wiring of the solderless breadboard, indicating the position of the components. Note the two three-pin connections for attaching the right and left motor control wires. These wires attach from the solderless breadboard (as shown) to the right and left motor control terminals on the L298 module. Be sure to observe correct polarity in the wiring, or else the circuit won't work. When mounting the photocells, don't clip off their leads. Leave them as long as you can. Add a length of heat shrink tubing, both to protect against short circuits and to stiffen the wire leads of the photocells. **Figure 13** shows the two photocells sticking out of the breadboard, with heat shrink tubing added. I selected the largest, most sturdy photocells I could find.

Once the control circuit has been built, secure the solderless breadboard to the breadboard deck using a 1" Velcro square. Secure the deck over the L298 module with a set of four 4-40" x 7/16" flat machine screws. Counter-sink the top holes of the deck so that the machine screws lay flat across the top. The deck fits neatly over the 1" standoffs in the four corners of the module.

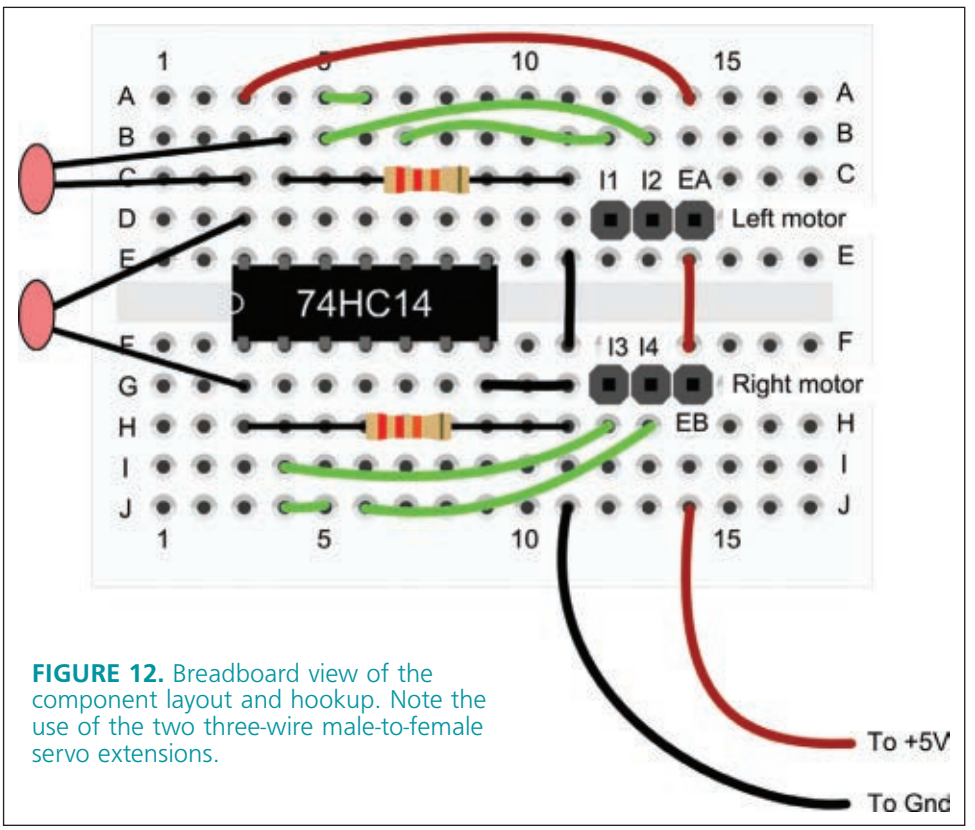


FIGURE 12. Breadboard view of the component layout and hookup. Note the use of the two three-wire male-to-female servo extensions.

Testing the Beginner Bot's Light Seeking Ability

Before inserting the batteries, double-check your wiring to make sure everything is correct. You'll

want to be especially careful about not mixing up the battery terminal wires. Never cross the + and – connections, or damage could result to the L298 module and any other electronics connected to the circuit.

Assuming all looks okay, insert the batteries and lift the wheels of the robot off the floor. Connect the battery clip to the battery holder. Both motors should immediately run. Point the two photocells to a bright light source, then observe.

Both motors should turn the same direction. If one motor turns in the opposite direction, remove power and flip the terminal wiring from that motor on the H-bridge.

Place the robot on the ground. The robot should move in the direction of the light. If it moves in the opposite direction, remove power and flip the wires for both motors.

With the Beginner Bot working and attracted to light, move to a darkened room. Bring along a flashlight, preferably one with a bright narrow beam. Apply power to the robot, and place it on the ground. With no or little light, the robot will travel in reverse. Shine the flashlight into the photocells. The robot should reverse direction and move toward the light.

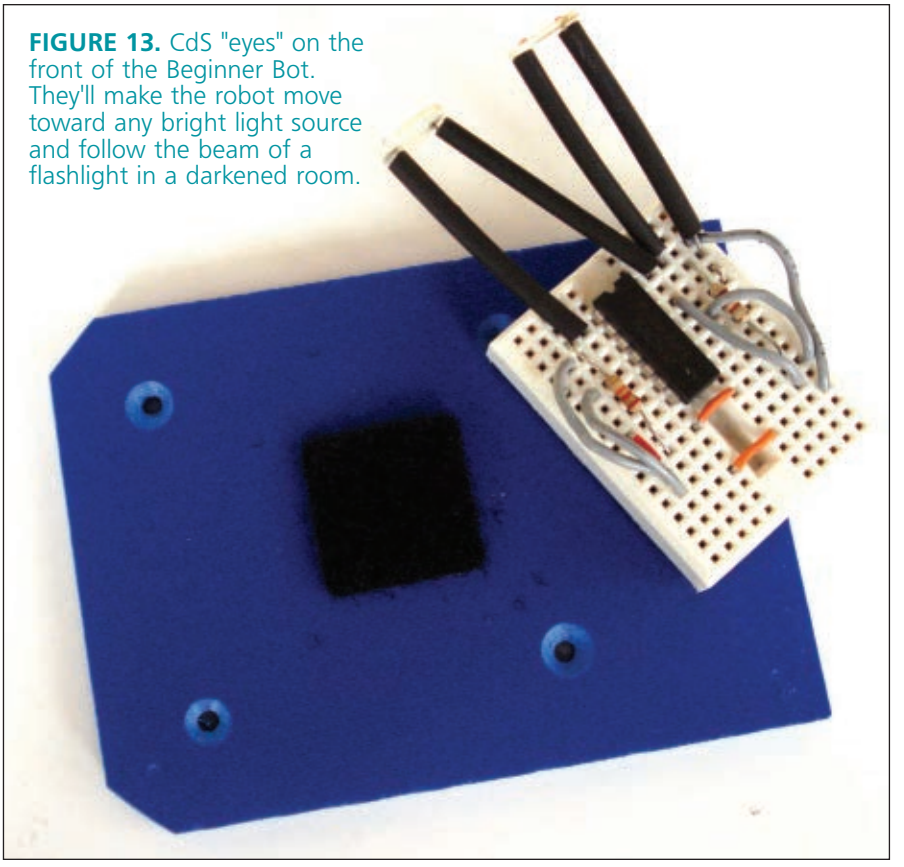


FIGURE 13. CdS "eyes" on the front of the Beginner Bot. They'll make the robot move toward any bright light source and follow the beam of a flashlight in a darkened room.

Get close to the robot and aim the flashlight into just one photocell (gently spread the cells apart if they're too close together). The robot should turn toward the photocell with more light.

Is your robot not responding to the light? Try operating the robot in a darker room. If the room is too bright, the robot will constantly move forward, and will be "blind" to the flashlight. Also try altering the value of the two resistors used with the photocells. Try higher or lower values to see what works best.

Next Up: Get Smart With Microcontrollers

There are lots and lots of ways to control your Beginner Bot with nothing more than simple electronics. For example, you can create a line following or maze solving robot using infrared sensors and LEDs, and some ordinary ICs. Or, you could add a bumper switch or two, along with a couple of LM555 timer chips and have your robot wander around the room, exploring its environment but reacting whenever it makes contact with an object.

I'll leave it up to you to explore these alternative types of circuitry, as now it's time to move on to the next phase: adding a miniature computer.

In the next installment, I'll demonstrate how to replace the light-sensing circuitry with a low cost microcontroller where you can define the operation of your bot by writing a few short lines of programming code. SV

Introducing the Cypress PSoC 5

by Lloyd Moore

www.servomagazine.com/index.php?/magazine/article/september2011_Moore

Cypress Semiconductor recently introduced the PSoC 5 processor and a line of development kits using it. For those not familiar with the Cypress PSoC line, the term PSoC stands for Programmable System on Chip, and it is what makes this particular line of processors unique. For many applications, additional hardware is not needed or is greatly reduced.

Not Just Another Processor

Figure 1 shows a simplified block diagram of the PSoC 5 processor. The PSoC 5 has the normal collection of on-chip resources you expect in a modern microcontroller such as: a processor core, Flash memory, RAM, counters, timers, analog-to-digital converters (ADC), and on-chip oscillators. In addition to the standard peripherals, there are blocks of software-configurable digital and analog resources. These are what make the PSoC processors unique.

The Digital Subsystem

The digital subsystem is where the PSoC family really shines. There is a small collection of dedicated hardware on the chip. Dedicated communications peripherals include: CAN 2.0, I^2C, and a full speed (12 Mbps) USB 2.0 transceiver. Four dedicated eight-bit timer blocks implement timers, counters, and PWM functionality. Each of the timer blocks is configurable for the specific function being performed. Two timer blocks can also be combined for 16-bit functionality.

The digital subsystem includes up to 24 Universal Digital Blocks (UDBs). UDBs are unique to the PSoC family and are used to implement all the rest of the digital peripherals typically found in a microcontroller. The easiest way to think about UDB blocks is to think of them as an arithmetic logic unit (ALU) of a conventional processor with a bank of programmable logic attached. **Figure 2** shows a block diagram of a UDB. At the top of the diagram is a block containing control logic for clock and status functions. Beside this are two blocks of programmable logic — essentially a PLD. Each of these blocks has the capacity to handle an eight product term Boolean expression. The programmable logic blocks can also be combined for more complex expressions.

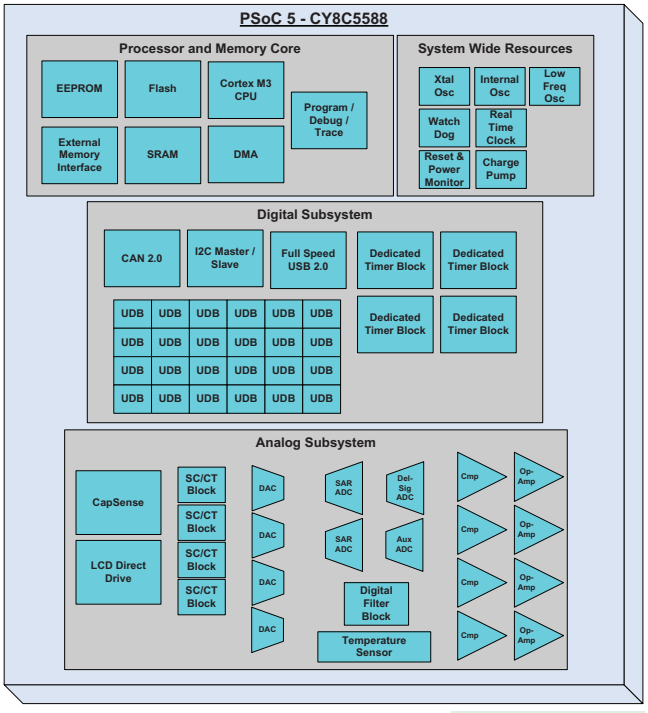


FIGURE 1.

The Datapath block contains a set of working registers along with a three-stage ALU. Everything in the Datapath is eight bits wide. First, the ALU stage performs one of the following operations: pass-through (NOP), increment, decrement, addition, subtraction, XOR, AND, or OR. Next, the shift stage performs one of four operations on the ALU result: pass-through, shift left, shift right, or nibble swap. Finally, the shifter result is masked and the final result is

passed back to one of the working registers. The Datapath block also includes two 4-entry-deep FIFOs. These allow data to be buffered between the UDB and processor core.

Like the fixed function timer blocks, the UDB blocks can be chained together to provide more complex operations than a single block can provide. This allows for 16-, 24-, and 32-bit operations in hardware. Additionally, the PLD blocks can be chained separately from the Datapath blocks. This allows resources to be used efficiently — even if the particular application does not directly map to the hardware architecture.

The Cypress development tool — PSoC Creator — provides a large library of pre-built components that can be easily added to your project. Among the existing peripherals are: a pseudo random number generator, UART, PWM generators, quadrature encoders, CRC generators, and shift registers. Finally, the PLD functionality can be used directly to implement any glue logic the design requires such as simple gates, flip-flops, or multiplexers. You will be able to see the full list a bit later on when we explore PSoC Creator. You can place as many components as you wish, in any combination, so long as there is physical hardware to implement them. Custom components can also be created.

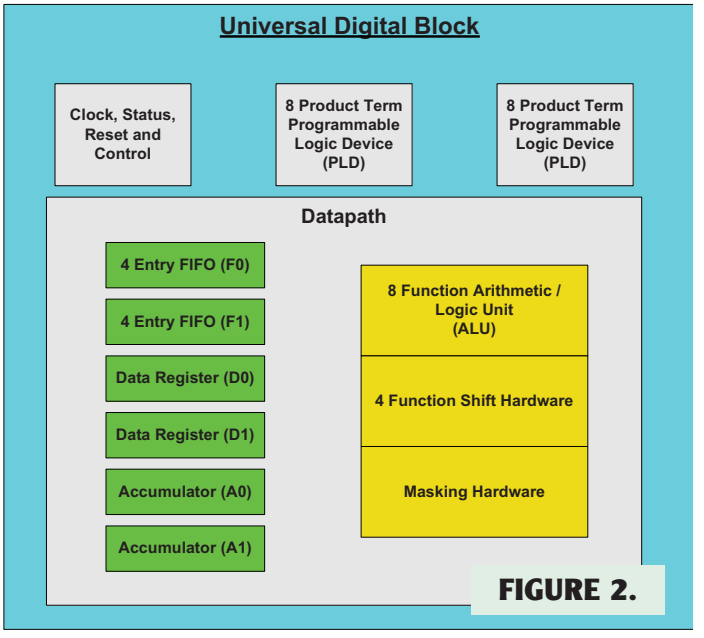


FIGURE 2.

The Analog Subsystem

The PSoC processor contains an analog subsystem. This portion of the processor is not as flexible as the digital subsystem, but equally valuable in eliminating external components from the design. Like the digital subsystem, there are fixed function blocks and programmable blocks. Among the fixed function blocks are four op-amps, four comparators, and a set of ADCs.

The specific ADCs vary from chip to chip. The largest PSoC contains four ADCs. Two of these are successive approximation devices able to do a 12-bit conversion at 1M samples/second. A resolution of eight or 10 bits can also be selected for this ADC, with a resulting increase in speed. A single delta-sigma ADC is also available. The delta-sigma ADC operates from eight to 20 bits of resolution with conversion rates from 384K samples/second at eight-bit resolution, to 10 samples/second at 20 bits of resolution. A fourth ADC — called the auxiliary ADC — is mostly used for reading on-chip levels such as the internal temperature sensor. All four ADCs are completely independent allowing for different configurations and concurrent operation. The Digital Filter block can be connected to the output of the ADCs to implement 24-bit FIR and IIR filtering algorithms on the data stream before the information is passed to the main processor, further freeing the main processor resources.

The analog system contains dedicated peripherals for handling a user interface. The LCD Direct Drive block allows for segmented or matrix-based LCD displays. On the largest chip, 736 segments can be driven in a matrix configuration. The CapSense block implements a variety of capacitive touch-based sensors. The block is composed of two separate subsystems allowing for combinations of discrete switches, switch arrays, or slider controls. Slider controls are implemented using multiple segments physically placed together. A slider control is able to approximate the finger position to a much higher resolution than the number of segments physically present. For example, five physical elements can be read as a 0 to 100 scale with one count increments.

Finally, the analog subsystem contains four programmable Switched Capacitor/Continuous Time blocks (SC/CT). Think of these as an op-amp with configurable resistors and capacitors in the input and feedback paths. These blocks can be used to implement programmable gain amplifiers, transimpedance amps, mixers, track and hold amplifiers, and, of course, more op-amps!

Just like the digital subsystem, pre-existing components are available which can be added to your project.

The Processor

The processor core is an ARM Cortex M3 which operates from DC to 80 MHz. This 32-bit processor includes an on-chip program, debug, and trace modules for development support. There is up to 256 KB of Flash memory, 64 KB of SRAM, and 2 KB of EEPROM memory available. Flash memory can also be programmed directly by the processor allowing for additional data storage or software updates in the field. Additional memory can also be added using the External Memory Interface.

Taking a Test Drive

Cypress currently sells a FirstTouch™ kit allowing you to experiment with the PSoC 5 processors. This kit has everything you need to get started, and also makes a great

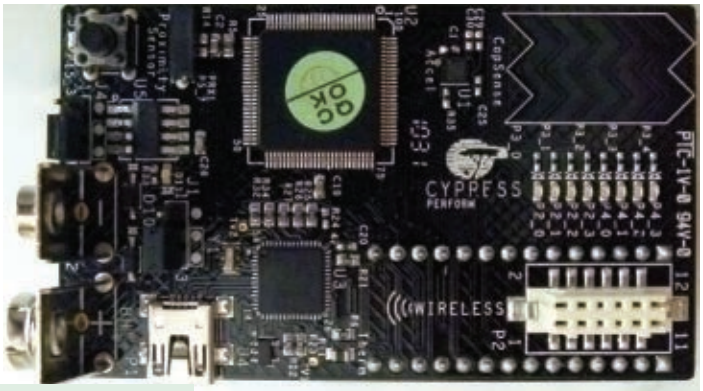


FIGURE 3.

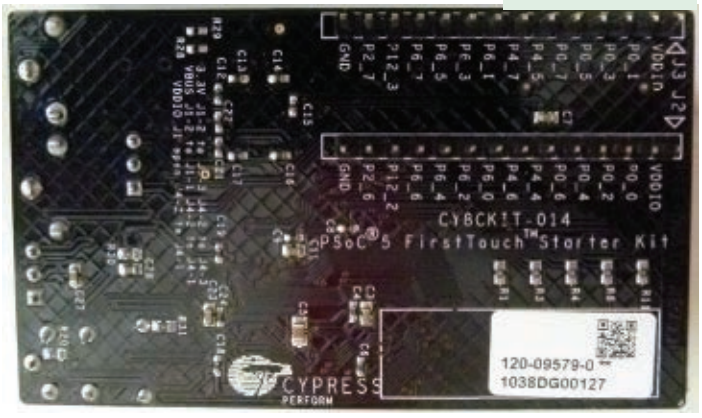


FIGURE 4.

controller for robotics. The kit is $49 and is available through the Cypress website at **www.cypress.com/go/CY8CKIT-014**.

The FirstTouch kit consists of a small board measuring 74 mm x 43 mm. **Figures 3** and **4** show the top and bottom of the board. There is a 28-pin DIP header on the bottom of the board for power and I/O. In addition to the PSoC 5 processor (CY8C5588AXI), the board also contains the following resources:

- Three Axis Accelerometer (Kionix KXSC7-2050)
- Thermistor (Murata NCP21XV103J03RA)
- CapSense Slider Pad (Five Elements)
- Momentary Pushbutton Switch
- Eight Red LEDs
- Voltage Regulator
- 9V Battery Connector

We'll install the development suite, called Cypress Creator. The tools are located on the CD included with the kit, or are available as a free download from **www.cypress.com/?rID=39551** as an ISO file titled "PSoC Creator 1.0 ISO image." Installing from the CD is the simplest route; simply load the CD. If you have "Autorun" enabled, the installer will start automatically. Once installation starts, run the program "cyautorun.exe" from the root of the CD. Clicking "Install PSoC 5 FirstTouch Starter Kit..." will install all of the tools and drivers needed. Simply follow the prompts accepting the default values.

Once you have the tools loaded, start up PSoC Creator and have a look around. Overall, the development environment is very similar to Visual Studio or Eclipse.

The CD includes six demo programs. These are located in the "Firmware" directory at the top level of the CD. Copy this directory to your hard drive so we can open the projects. Next, open the "ProximitySense" demo in PSoC Creator by using the "File -> Open -> Project/Workspace..." menu. The file you want to open is named "ProximitySense.cywrk;" "cywrk" is the extension for a Cypress workspace file.

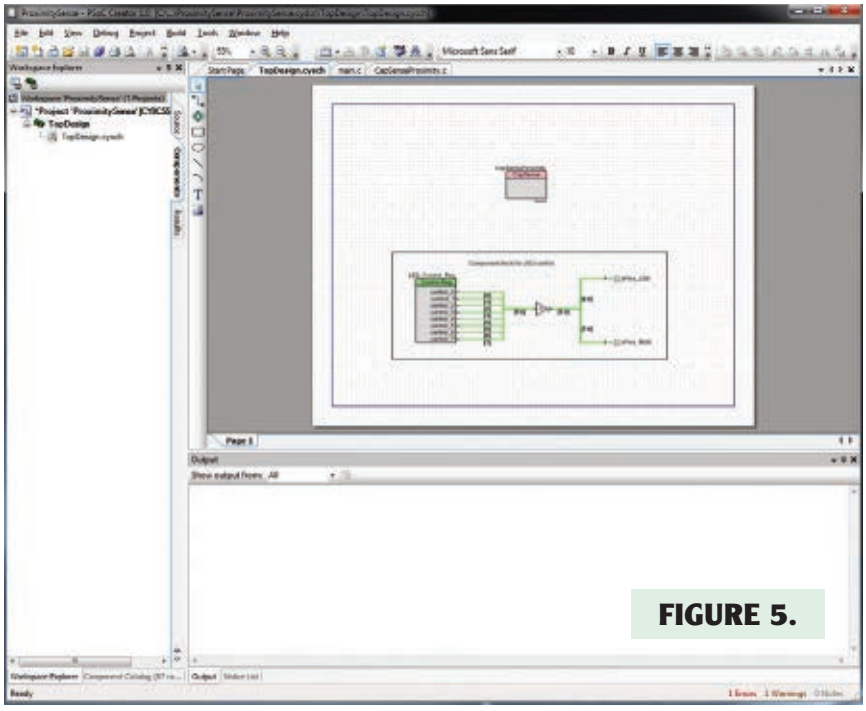


FIGURE 5.

If you have a newer version of PSoC Creator, the demos created with the "Component Update Tool" will be launched. This tool automatically updates the pre-built components used in a project to the latest versions. If you see this tool come up, click on the button "Update All to Latest" and follow the prompts. You will want to do this to update each of your projects when a new version of Creator is released.

Once you have the workspace opened, it should look similar to **Figure 5**. If the "TopDesign.cysch" tab is not available, you can open it by clicking on the filename in the "Workspace Explorer." The "cysch" file is basically a schematic capture where you draw the circuitry to be implemented in the programmable hardware portions of the PSoC. With the schematic visible, on the lower left you will see a tab titled "Component Catalog." Clicking on this tab

Cypress PSoC 5 FirstTouch™ Starter Kit
Mfg Part Number: CY8CKIT-014
Price: $49 USD
Available at **www.cypress.com/go/cy8ckit-014**

will open the collection of pre-built components. Clicking one of the components in the catalog brings up a preview of the schematic symbol and a link to the component datasheet in the lower left-hand corner of the window. You can also right-click on a component to access the datasheet, and locate an example project showing the component being used. Components are added to a design by dragging them from the Component Catalog on to schematic. Once in the schematic, you can double-click them to open up a configuration dialog for that specific instance of the component. The "Wire Tool" will allow you to make electrical connections between components, just like you are drawing a schematic. This tool is available on the left edge of the schematic window and appears as a line with boxes at the end.

Open up one of the component datasheets by either right-clicking on the component and selecting "Open Data Sheet..." or clicking on the link in the lower left corner of the window. Component datasheets look very similar to datasheets for integrated circuits. There is a general description of the component, how to wire the component, and how to set the various options of the component. In addition, PSoC components also have a 'C' level API that is automatically generated by the tool chain for each component. Each API is component specific, but there are two naming conventions worth noting. Every component has a XXX_Start() function and XXX_Stop() function. These functions initialize and shut down the component. When writing software for the PSoC, you must call the XXX_Start() API for every component in use, even if it will operate completely independent from the processor from that point on.

In the "Workspace Explorer," double-click on the file "Proximity Sense.cydwr." This opens the Design-Wide Resources Editor. From here, you will map nets from your schematic to specific pins on the processor, configure clocks, interrupts, and other system resources. Drag a pin alias on the right-hand side of the window to a specific pin on the processor to connect the two together.

The PSoC processor is programmed in C, and the PSoC 5 uses the GCC tool chain. The Cypress folks have gone to great lengths to preserve a standard C environment. The proper way to access any component is through its API. Additionally, the processor itself has standard symbols, defines, and API routines. It is particularly important to use the standard APIs on the PSoC because the location of specific hardware can move from build to build. Additionally, when an upgrade is available, only the API is guaranteed to be preserved. All other aspects of a component can and do change. Let's program the ProximitySense demo into the FirstTouch™ board. Plug the USB cable into your computer and then into the mini-USB connector on the board. The drivers will load automatically. Wait a few moments for this process to complete.

Now, select the "Debug -> Program" menu item. Since the project hasn't been built yet, this will automatically start a compilation before programming the chip. Assuming all goes well, the build will complete and programming will start automatically. When everything is finished, you will see the message "'PSoC5 CY8C5588AX*-060' was successfully programmed at..." in the Output window.

Playing with CapSense

Unplug the board from the USB cable. Inside the development, kit you will find a 9V battery and a small wire. Plug the wire into the socket on the board marked "Proximity Sensor." This serves as the antenna for the CapSense proximity sensor. Plug in the 9V battery and wave your hand over the antenna wire. The LEDs will light up as you get close to the wire.

At this point, you can take the board around and see how it responds to various materials. Large objects will give some level of response. Smaller objects will also give a response depending on how conductive they are. More conductive objects will have a larger response. Very thin, non-conductive objects have the most problems being detected.

What Next?

I particularly like the ProximitySense demo as it provides the basis for creating a non-contact object detection system. There are many ways this demo can be enhanced to replace bump switches. Furthermore, the level and pattern of a response can give you clues about the object itself. Tuning the sensor is beyond the scope of this article; however, the CapSense component is highly configurable. Different component settings and configurations of antennas can also be used.

The connectors on the bottom of the FirstTouch board (J2 and J3) are designed to fit into a 28-pin DIP socket. This gives you a simple way to connect the FirstTouch board to your own projects. Power can be supplied through this connector instead of using the 9V battery contacts. P2 on the top of the board can also be used for additional I/O.

Finally, Cypress has a series of on-demand training videos available at: **www.cypress.com/?id=2233&rtID=134**. The sessions are numbered like college courses and should be viewed in order. The Cypress website has a large collection of datasheets, application notes, code examples, and other resources to support the PSoC processors. SV

Lloyd Moore has worked as a software and hardware engineer in the fields of robotics, machine vision, and industrial automation for 25 years. Presently, Lloyd is the president of CyberData Corporation which develops custom automation solutions worldwide.

UPGRADING THE BOE-BOT

PART 1

by William Henning

For those of you who are not yet familiar with the Boe-Bot, it is a small (approximately 5" x 6") robot with two wheel drive and a small spherical ball acting as a third wheel. You steer the Boe-Bot by controlling the speed and direction of the two continuous rotation servos. Due to the differential drive, Boe-Bot can turn on a dime and even spin in place!

[The Parallax Boe-Bot is probably the most successful robot for the educational market in North America – and it is also very popular with hobbyists. Recently, the Boe-Bot became a recommended platform for completing the Boy Scouts of America Robotics Merit Badge.]

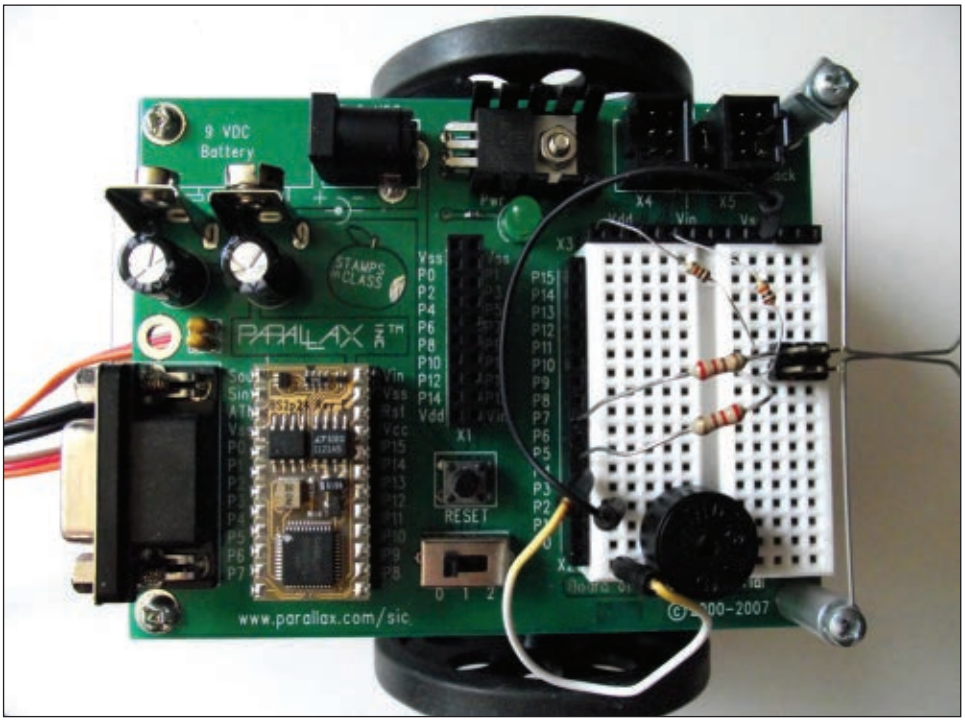


FIGURE 1. Board of Education mounted on the Boe-Bot chassis.

The standard Boe-Bot kit uses a Parallax Board of Education (BOE) as the "brain" of the robot, and the BASIC Stamp processor (BS2) provides a total of 16 I/O lines to experiment with — four of which go to three-pin servo connectors.

The excellent *Robotics with the Boe-Bot* book initially configures Boe-Bot with two "whiskers" (basically, momentary contact switches) to sense if it runs into anything. Later experiments use the included two IR LEDs and 38 kHz IR detectors as distance sensors so Boe-Bot won't have to bump into objects.

The BASIC Stamp used on the Board of Education does not have any "real" analog-to-digital converters (ADC) in it. It does include an "RCTIME" command that can be used to measure the time required to discharge a capacitor. However, that can take many milliseconds — time that cannot be

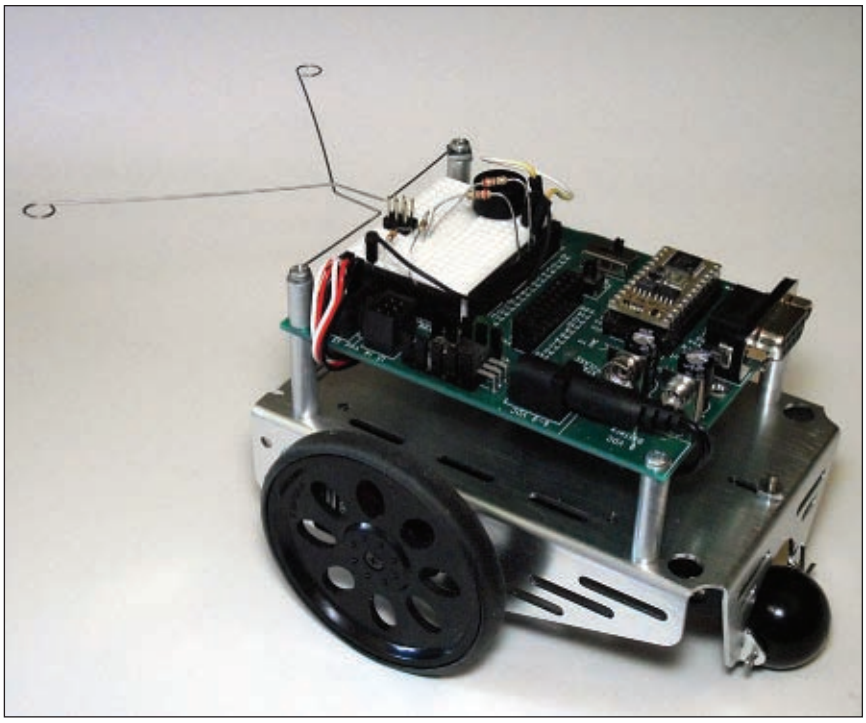


FIGURE 2. *Robbie with whiskers.*

used to do other processing. For best results, the whole event loop of a program should complete in just over 20 ms, as otherwise the servos won't be updated frequently enough. I love tinkering with my Boe-Bots. They are a great platform for experimenting with robotics. However, I hankered for more sensors, more processing power, and longer battery life – basically more capabilities to explore robotics with.

Before I start describing my hack, **Figure 2** is a photo of "Robbie" equipped with only whiskers and a piezoelectric speaker. You can compare it later with upgraded shots.

PARALLAX UPGRADES FOR BOE-BOT

There are some easy upgrades for the Boe-Bot straight from Parallax:

1) BS2p24 with ~3x speed and 8x the program memory of the BS2, 128 scratchpad variables and 19 more PBASIC commands. There is also the BS2px for about 50% more speed than the BS2p24. Watch out for changes in RCTIME, PULSIN, and PULSOUT.

2) The Boe-Boost module adds a fifth AA cell, boosting the voltage of the power supply from 6V to 7.5V if you are using alkaline batteries. You can boost the voltage from 4.8V to 6V if you are trying to use Ni-MH batteries.

3) A Ping))) sensor and bracket provide a way for Boe-Bot to tell how far it is from the nearest object in line with the direction the Ping))) sensor is pointing. The Ping))) sensor is often mounted on a standard servo, so it can sweep side to side, thus find any objects in a 180° arc in front of the Boe-Bot at distances up to three meters away.

I must admit that the BS2p24 quickly found a home on Robbie. I couldn't resist three times the speed and eight times as much space for programs and 128 scratchpad variables. The extra PBASIC 2.5 commands didn't hurt, either.

Unfortunately for my wallet, that processor upgrade was only a stop-gap measure, as I really wanted a LOT longer battery life, more processing power, more memory, and an ADC with multiple input channels for Robbie.

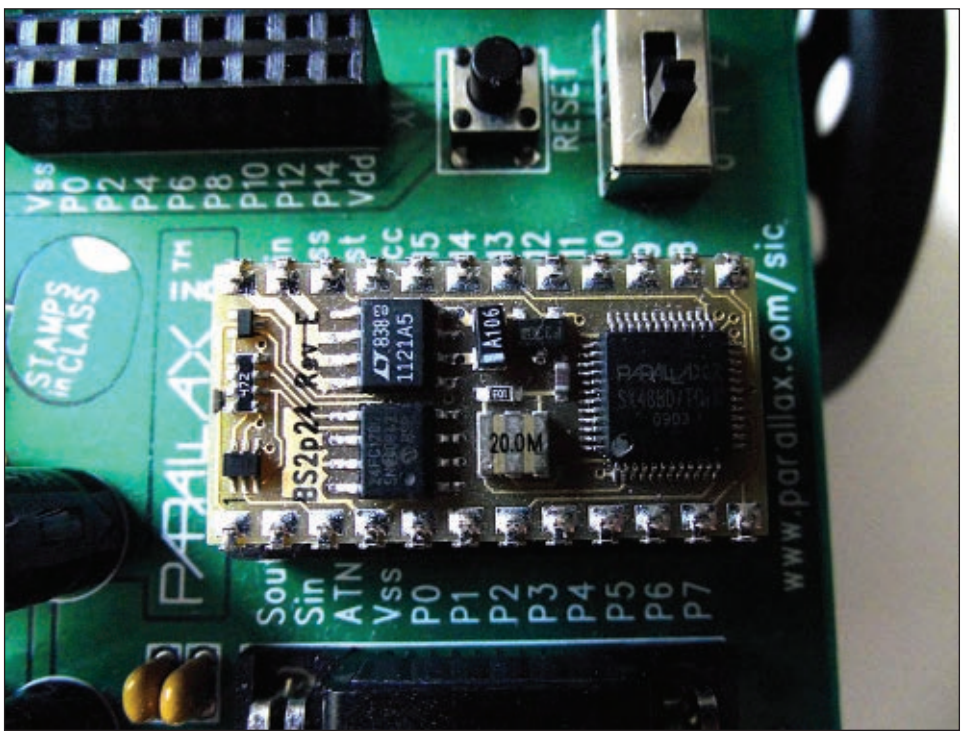


FIGURE 3. BS2p24 BASIC Stamp on the Board of Education.

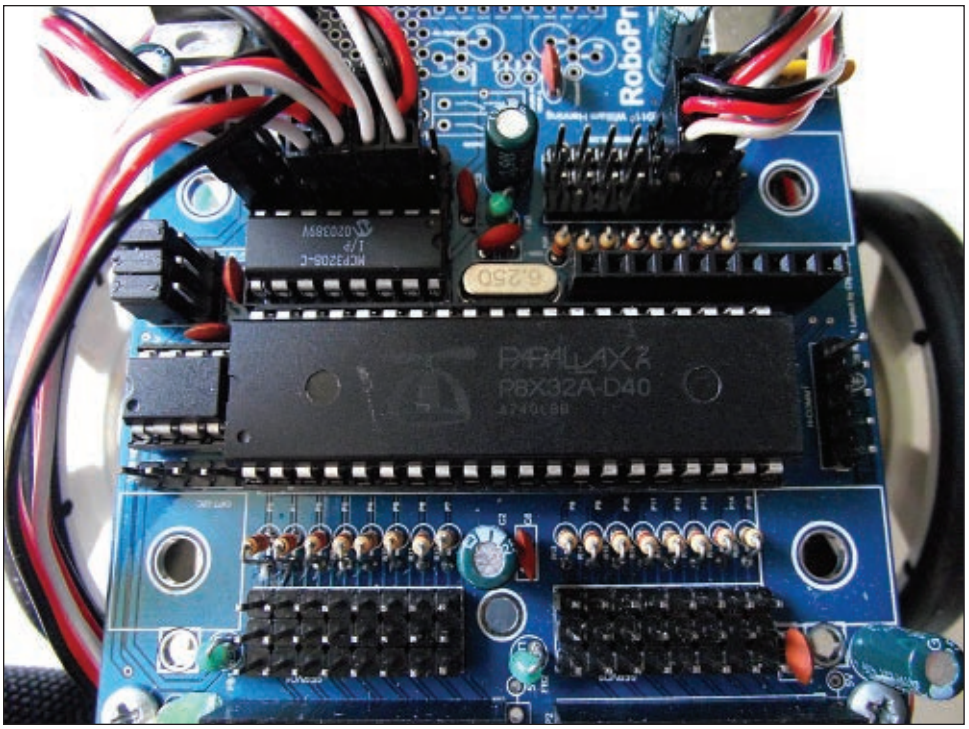


FIGURE 4. Close-up of the Propeller on RoboProp.

holder I had mounted on top of the chassis. At least I could use six Ni-MH batteries instead of using disposable alkaline batteries.

Okay, that solved my battery life problem. Now, what do I do about getting more processing power, memory, and analog inputs?

UPGRADING FROM THE BASIC STAMP

Parallax manufactures a much more powerful and advanced processor called the Propeller – which has eight separate 32-bit RISC processors that share 32 KB of RAM. Each cog (what Parallax calls the cores) also has 2 KB of combined code/data space, for a total of 48 KB of static RAM in the processor. The Propeller is available in TQFP, QFN, and nice breadboard-friendly DIP40 packages, and Parallax has a number of prototyping boards for it. I have been working with the Propeller since its introduction in 2006, and I find it to be an excellent "swiss army knife" for embedded development. The Propeller boots from the first 32 KB in an EEPROM, loading it into the 32 KB shared RAM. Then, cog0 automatically starts executing the Spin program in the hub (shared RAM). Spin is the high level language designed specifically for the Propeller, however, Spin is also capable of starting Assembly language programs in the cogs. Basic, C, and Forth are available for the Propeller, as well.

My first Propeller-based robot – Johnny – used the serial Propeller proto board for its brain.

UPGRADING THE BATTERIES – TAKE ONE

Instead of getting a Parallax Boe-Boost, I changed the four AA cell battery holder for a pair of three AA cell holders wired in series. This boosted the voltage to about 7.2V with Ni-MH batteries, and 9V with alkaline batteries significantly improving battery life. Unfortunately, it was awkward to change all six batteries; I had to take the BOE off the bot whenever I changed the three cells in the

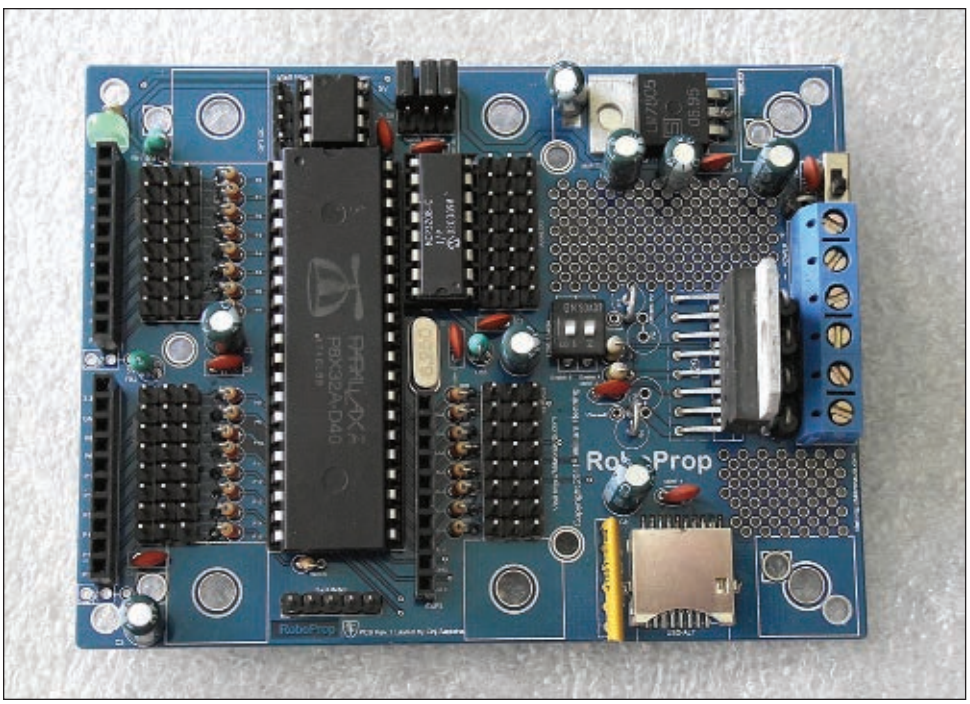


FIGURE 5. RoboProp with L298 dual H-bridge for driving DC motors.

ROBOPROP

For Robbie, I decided to upgrade the Boe-Bot with my commercial Propeller based

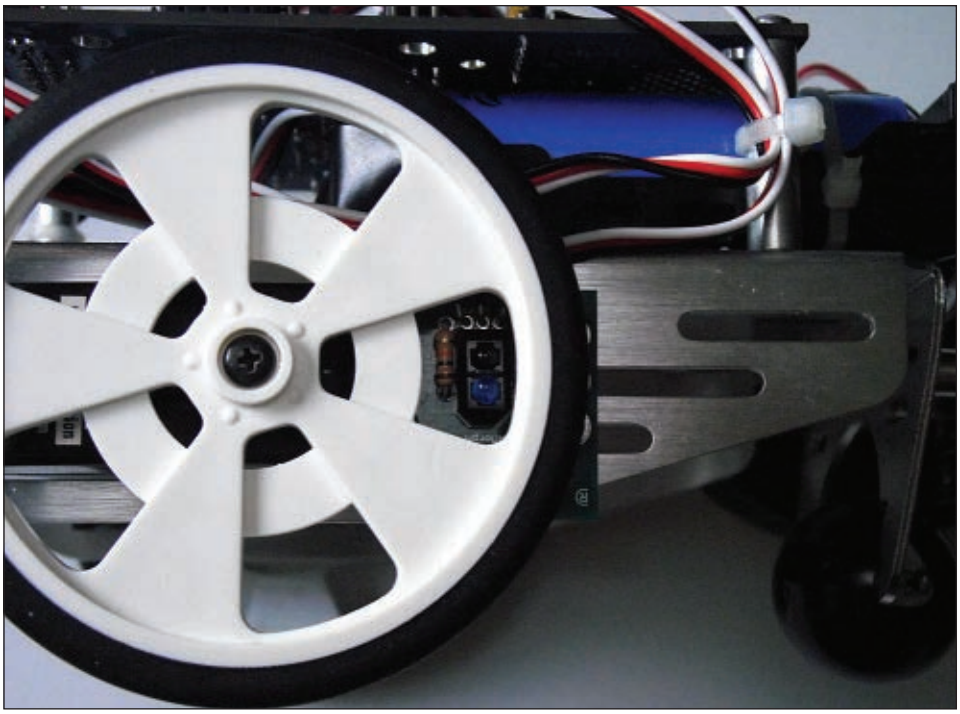


FIGURE 6. Wheel encoder mounted on the RoboProp-Bot chassis.

RoboProp advanced robot controller board, available from my website at **http://Mikronauts.com**.

RoboProp has:

- A Propeller with eight 32-bit 25 MIPS processor cores, running at 100 MHz.
- 64KB of EEPROM.
- 24 general-purpose I/Os that can be used for servos or digital inputs.
- Eight 12-bit analog inputs for sensors.
- Four of the I/Os can be used for the optional on-board L298 H-bridge to drive DC motors.
- Four are used for the on-board micro SD card socket if a µSD card is inserted (up to 32 GB).

Replacing the Board of Education with RoboProp is quite simple:

- Disconnect all wiring going to the BOE.
- Remove the four mounting screws.
- Remove the BOE from the standoffs.
- Put RoboProp where BOE used to sit.
- Screw in the four mounting screws.
- Decide on where to re-attach all the wiring.

I used the following wiring map:

- Red and black leads from the battery pack to the screw terminal + and -, respectively.
- Left servo to P18 on RoboProp.
- Right servo to P19 on RoboProp.

UPGRADING THE BATTERIES – TAKE TWO

Since I found it awkward to change the six Ni-MH batteries from my previous power supply upgrade, I decided to switch to non-AA batteries. After some research, I settled on the 18650 Lithium-Ion batteries. These cells look like somewhat overgrown AA cells, and provide 3.7V at up to 4,000 mAh, depending on the cells you buy.

I took off the two three-cell AA holders and installed a three-cell 18650 holder beneath the PCB on the metal bot chassis in such a way that I could pull out the holder to change batteries without dismounting the controller board. When fully charged, this gave 11.1VDC which is dropped to 10.6V by the reverse voltage protection diode on RoboProp.

ADDING WHEEL ENCODERS

Unfortunately, the Boe-Bot does not have wheel encoders as it ships, so it was not easy to get Robbie to travel in a straight line.

The least expensive workaround is calibrating the motor speeds by measuring the deviation from traveling in a straight line over some distance, and then reducing the speed of the faster motor to match the speed of the slower one. This allows the robot to travel in a straighter line than it could without such calibration.

A better solution is to add wheel encoders, so the robot can measure how much the wheels have turned.

Parallax has a Boe-Bot Digital Encoder Kit which measures the rotation of the wheels using a reflective IR sensor that can sense the openings in the standard Parallax wheels. The accompanying documentation is excellent, and shows how to implement a wheel encoder based calibrated dead-reckoning navigation system using the 16 pulses per revolution obtained from the reflective IR encoders.

I decided to use a different approach for this upgrade. I used my small servo mounted "SirMorph" IR short range distance sensors for wheel encoders. SirMorph was designed to mount on standard size servos for use with Solarbotics 2-5/8" servo wheels, and provides 10 pulses per revolution.

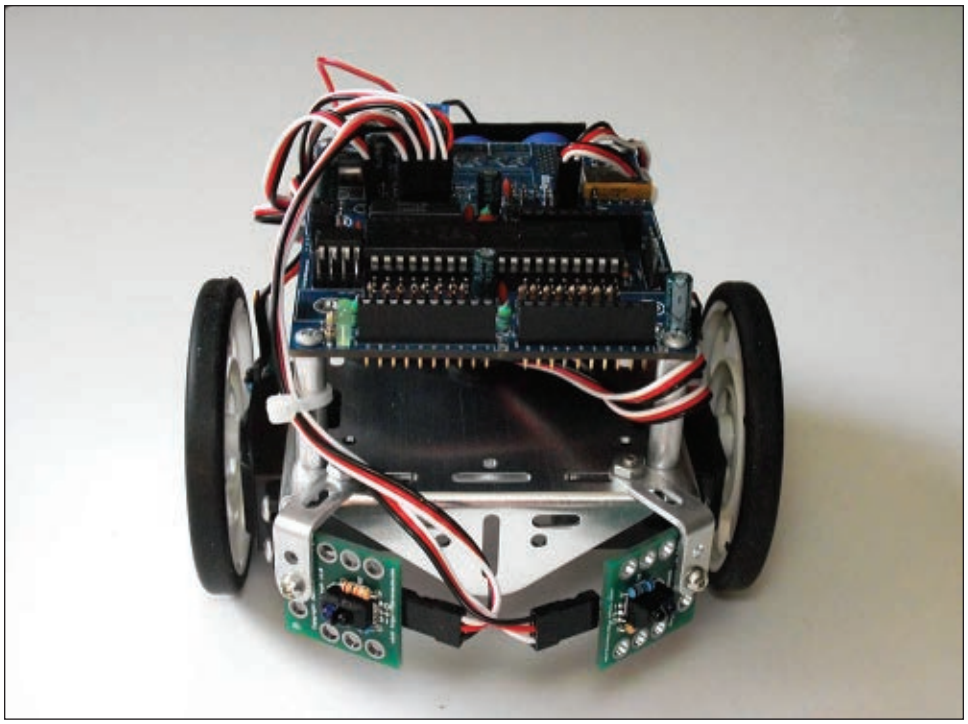


FIGURE 7. Front IR bumpers mounted on the RoboProp-Bot.

SirMorph provides an analog output; it can also be used as a virtual bumper, short range IR distance measurement device, or line following sensor, which makes it more versatile. As SirMorph was designed for the Solarbotics series of servo wheels, I am also replacing the standard Parallax wheels with white Solarbotics wheels to accomodate SirMorph. For the wheel encoders, I used:

- Left wheel encoder on AN1.
- Right wheel encoder on AN2.

This allows you to write code to:

- Travel a specific distance.
- Calculate the RPM at a given servo speed.
- Perform dead-reckoning navigation.

The wheel encoders I am using can detect 1/10th of a rotation of the 2-5/8" wheel. Therefore, 1/10th of a wheel rotation travels (pi * 2.625")/10 inches — that is, approximately 0.825". A full rotation travels 8.25".

You can calculate the RPMs achieved at a given servo speed by counting the number of edge transitions in one minute and dividing by 10; if lower precision is acceptable, just count the number of edges in six seconds for a direct RPM reading.

The RoboProp library "GetRPM(servo_pin, analog_input, speed)" function returns the rotations per minute for the servo, encoder, and speed as an integer.

You can use GetRPM to make a table of RPMs corresponding to different servo speed settings which will allow you to travel in straighter lines than before, generating a calibration table simply by using the servo speed you need for each wheel to get the same RPM, then running both servos at the same speed for the same amount of time.

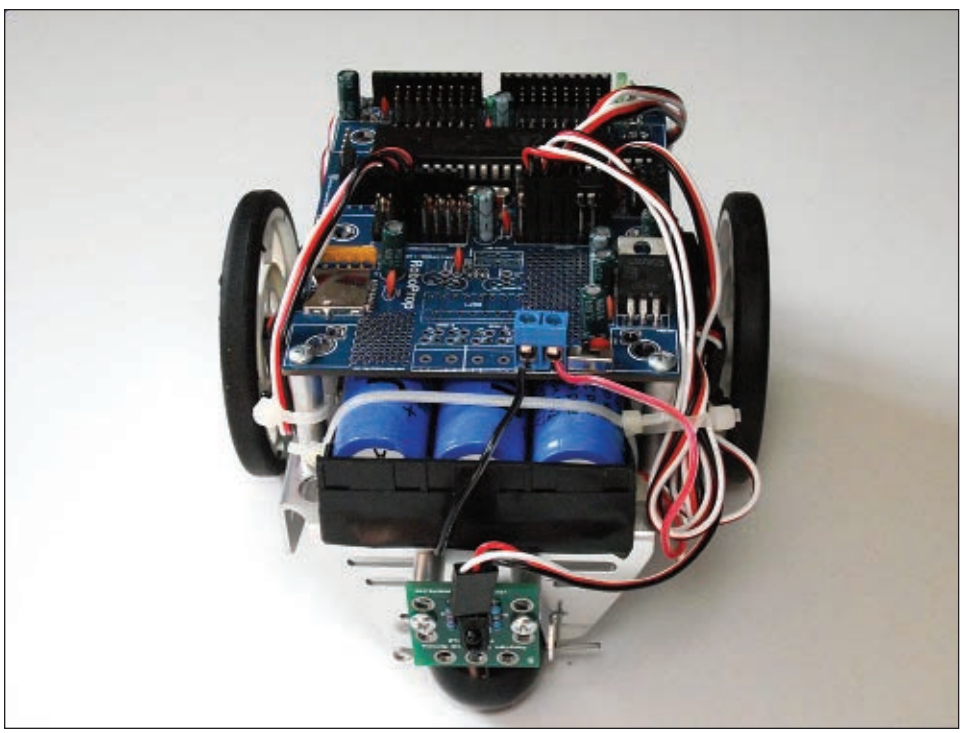


FIGURE 8. Rear virtual bumper and three 18650 batteries.

REPLACING THE WHISKERS

For the simple Boe-Bot roaming demo program, I used the Parallax-provided whisker contact sensors to make Robbie turn to avoid obstacles, and to back up if he ran straight into something.

I decided to replace the whiskers with short range IR

```
' Boe-Bot PBASIC Roaming code

DO

  IF (lwhisker = 0) AND (rwhisker = 0) THEN
  ' can't go forward
    'DEBUG "STUCK!",13
    GOSUB backup
    GOSUB left
  ELSEIF lwhisker = 0 THEN ' can't go left
    'DEBUG "LWHISKER",13
    GOSUB right
  ELSEIF rwhisker = 0 THEN ' can't go right
    'DEBUG "RWHISKER",13
    GOSUB left
  ELSE ' go forward
    'DEBUG "FORWARD",13
    GOSUB forward
  ENDIF

LOOP
' RoboProp-Bot Spin Roaming code

repeat

  if left_bumper AND right_bumper
    ' can't go forward
    'robot.debug(string("STUCK!",13))
    backup(2000)
    left(1000)
  elseif left_bumper ' can't go left
    'robot.debug(string("LBUMPER",13))
    right(50)
  elseif right_bumper ' can'tgo right
    'robot.debug(string("RBUMPER",13))
    left(50)
  else  ' go forward
    'robot.debug(string("FORWARD",13))
    forward(50)
```

LISTING 1. PBASIC on Boe vs. Spin on RoboProp.

```
' helper function for checking the state
' of the left bumper
pub left_bumper
  return robot.ain(lbumper) > BUMPER

' helper function for driving forward
pub forward(n) ' most common movement, falls
into move
  'robot.debug(string(13,"FWD",13))
  move(n,speed,speed)
```

Simple helper functions make it easier to read the code for beginners and advanced users alike.

Note that the debug serial output is commented out in all of the examples shown.

LISTING 2. Sample Spin helper functions.

```
' Roaming mode

repeat

  ' go faster if chased
  if rear_bumper
    'robot.debug(string("GO FAST",13))
    speed := 300
  else
    speed := 50

  if left_bumper AND right_bumper
    ' can't go forward
    'robot.debug(string("STUCK!",13))
    backup(2000)
    left(1000)
  elseif left_bumper ' can't go left
    'robot.debug(string("LBUMPER",13))
    right(50)
  elseif right_bumper ' can'tgo right
    'robot.debug(string("RBUMPER",13))
    left(50)
  else  ' go forward
    'robot.debug(string("FORWARD",13))
    forward(50)
```

Here is what the main body of code looks like after adding support for the rear virtual bumper.

Robbie sprints away when any object approaches too closely from behind.

That was easy! Just imagine how much fun your cat would have chasing a robot that keeps running away from him.

LISTING 3. Roaming mode with rear sensor watching for pesky cats.

sensors, and I also decided to add another sensor to boot: a rear facing short range IR sensor. The three new IR sensors act like virtual bumpers, allowing Robbie to react to his environment, and back out of corners.

I added the new sensors as follows:

- Left front virtual bumper on AN3.
- Right front virtual bumper on AN4.
- Rear virtual bumper on AN7.

To demonstrate Robbie's new capabilities, I wrote a more sophisticated version of my Boe-Bot roaming code in

Spin, using the RoboProp library.

Check out the main body of the RoboProp-Bot roaming code, side by side with the Boe-Bot version in the **Listings** included here.

As you can see, the code is very similar; the greatest difference is how the bumpers are handled.

On the BOE, the whiskers use digital inputs declared as PIN variables which were used directly in the expressions. The BASIC Stamp does not support procedures or functions, so the movement commands had to be written as subroutines invoked by the GOSUB keyword. The servo speed is controlled by global variables, as is the time to travel.

On RoboProp, the analog short range IR sensors were connected to analog inputs, read using a function call, and compared to a constant BUMPER distance. Had I used whiskers, I could have written

```
"if ina[lwhisker] == 0 AND ina[rwhisker] == 0"
```

Initially, I wrote:

```
"if (robot.ain(lbumper) > BUMPER) AND
(robot.ain(rbumper) > BUMPER) "
```

However, to make it easier for readers, I added some helper functions allowing me to write:

```
"if left_bumper AND right_bumper"
```

Spin supports methods with arguments (think functions and procedures), so I passed the "time to travel" (in milliseconds) at the current speed as a parameter to the helper forward/backward/left/right methods. I could have also passed the speed, but I decided to use a global variable for that purpose.

Take a look at **Listing 3** to see what the main body of code looks like after adding support for the rear virtual bumper. Robbie sprints away when any object approaches too closely from behind.

I encourage you to download the full source code for both versions and compare them. You will see how easy it is to program robots in Spin.

COMING NEXT

In Part 2, Robbie gets more sensors, greatly expanding his capabilities! SV

REFERENCES

Robotics with the Boe-Bot, Parallax Inc., ISBN: 1-928982-03-4.

BASIC Stamp Syntax and Reference Manual v2.2, Parallax Inc., ISBN: 1-928982-32-8.

Propeller Manual v1.1, Parallax Inc., ISBN: 9781928982470.

Getting Started With FPGAs — Part 1

by David Ward

www.servomagazine.com/index.php?/magazine/article/september2011_Ward

You've probably heard about FPGAs (field programmable gate arrays) recently if you're involved in the electronics field. FPGAs are some of the latest ICs to be found in electronics products such as: digital flat screen televisions, digital cameras, and multi-function printers. Anyone involved in the field of digital electronics should be interested in learning more about FPGAs because of their growing influence in the field of electronics.

Since most FPGAs are bundled in surface-mount packages, it isn't very practical to try and breadboard one to experiment with. There are several good FPGA training boards available from Digilent; go to **www.digilentinc.com** to see what they have available in FPGA trainers.

This two-part series FPGAs will deal with the Digilent Basys2 FPGA board which costs from $49 up to $79 depending on whether you are a student or not (see **Photo 1**). The Basys2 FPGA board comes with a Xilinx XC3S100E 100K gate FPGA or, for another $20 you can upgrade the board to a XC3S250E 250K gate FPGA. This article will only deal with the basic 100K FPGA.

You will probably want to purchase four of the 6" six-pin expansion cables for $2.50 each if you want to expand your design out to an external bread-board, which will be

PHOTO 1.

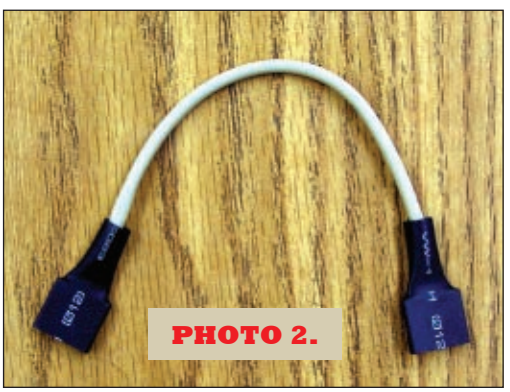
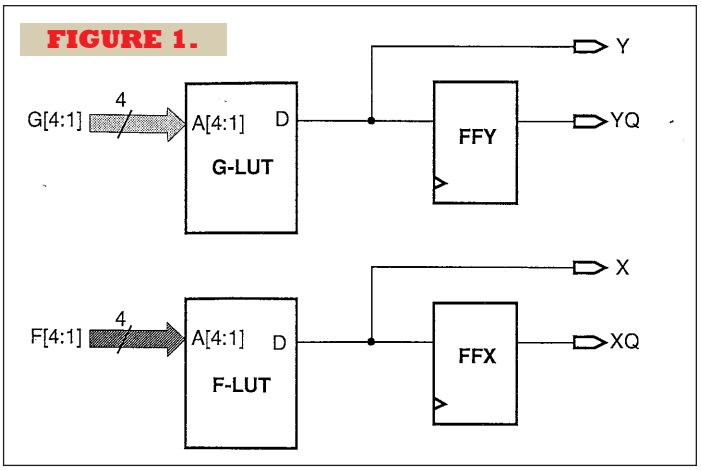
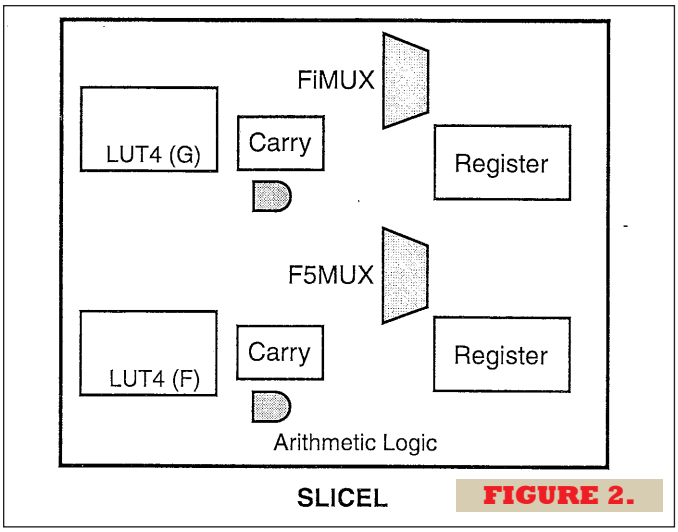
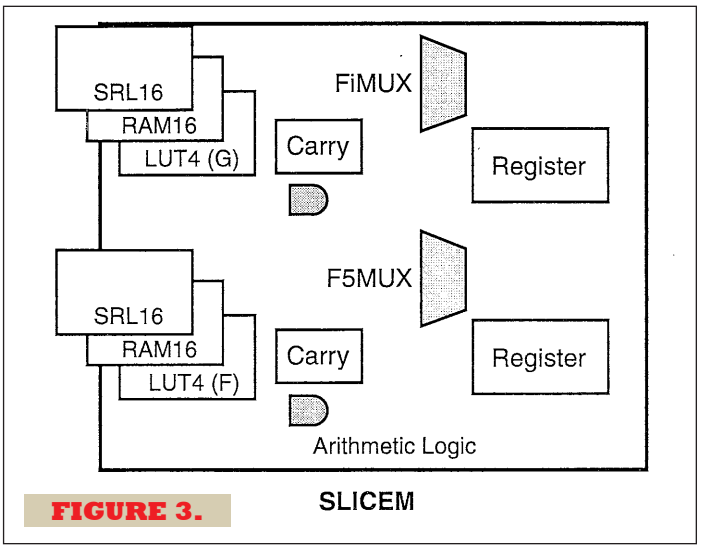


PHOTO 2.







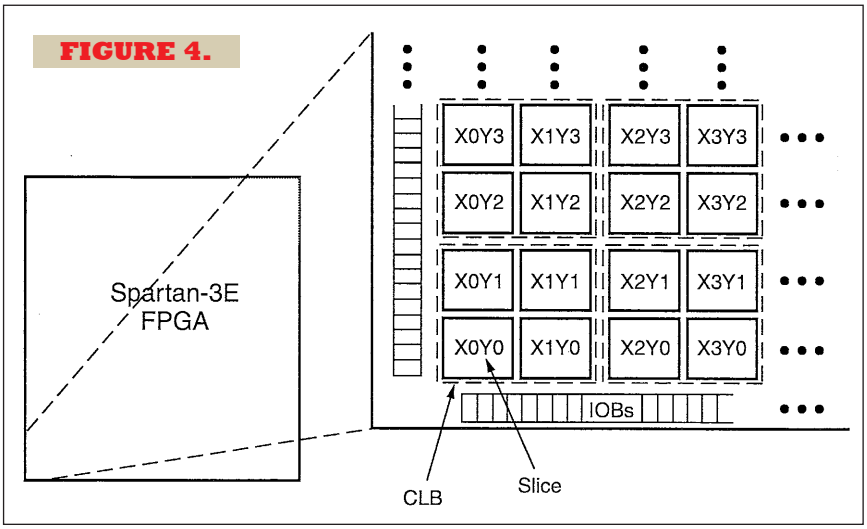
demonstrated later on (see **Photo 2**). The Basys2 FPGA board comes equipped with eight slide switches, four pushbuttons, eight LEDs, four seven-segment displays, a VGA connector, a PS2 connector, and a USB connector. It is powered by your PC's USB connection or from an external 3.5V to 5.5V DC source.

FPGAs are volatile devices. That is, their configuration bit file – the file that tells them how to make their internal connections to perform the functions that you desire – is lost each time the FPGA is powered down. Therefore, FPGAs must have an external non-volatile memory device to hold this configuration bit file. The Basys2 FPGA board has a 2M bit Xilinx XCF02 non-volatile platform Flash PROM (programmable read only memory) chip on-board for this purpose. However, it is possible to download a configuration bit file from your PC directly into the FPGA and have it run as long as power is connected to the board. If you want your configuration bit file to operate after powering the board down, you will need to load that file into the PROM instead.

Let's look at the internal structure of the Xilinx XC3S100E FPGA. At the most fundamental level in the FPGA is the LUT (look-up table). Each LUT in the FPGA has four inputs and one output, and any four variable Boolean logic operations can be implemented into one LUT. If more inputs are needed, the LUTs are cascaded together. There are "G" LUTs and "F" LUTs; see **Figure 1**.

Two LUTs, one G LUT, and one F LUT are combined together to make what is called a SLICEL, as shown in **Figure 2**. When two LUTs are combined together with a 16-bit RAM register, it is called a SLICEM; see **Figure 3**. There are a total of 1,920 LUTs in the FPGA: 960 G LUTs and 960 F LUTs. Since SLICEMs have a 16-bit RAM register, they make up a total of 15,360 RAM bits; 960 x 16 = 15,360. This is what is referred to as distributed RAM in the FPGA. These 16-bit SLICEM registers can also be used as shift registers.

Four slices, two SLICELs, and two SLICEMs are interconnected to make a CLB (configurable logic block). There are 240 CLBs in this FPGA. CLBs are placed in rows and columns; 22 rows by 16 columns for a total of 240 as shown in **Figure 4**. Notice



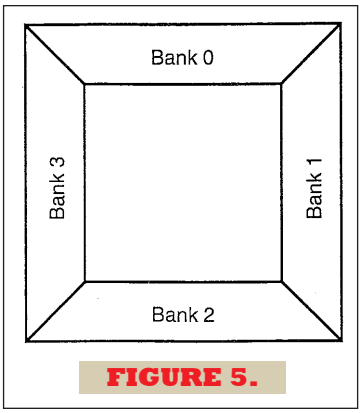


FIGURE 5.

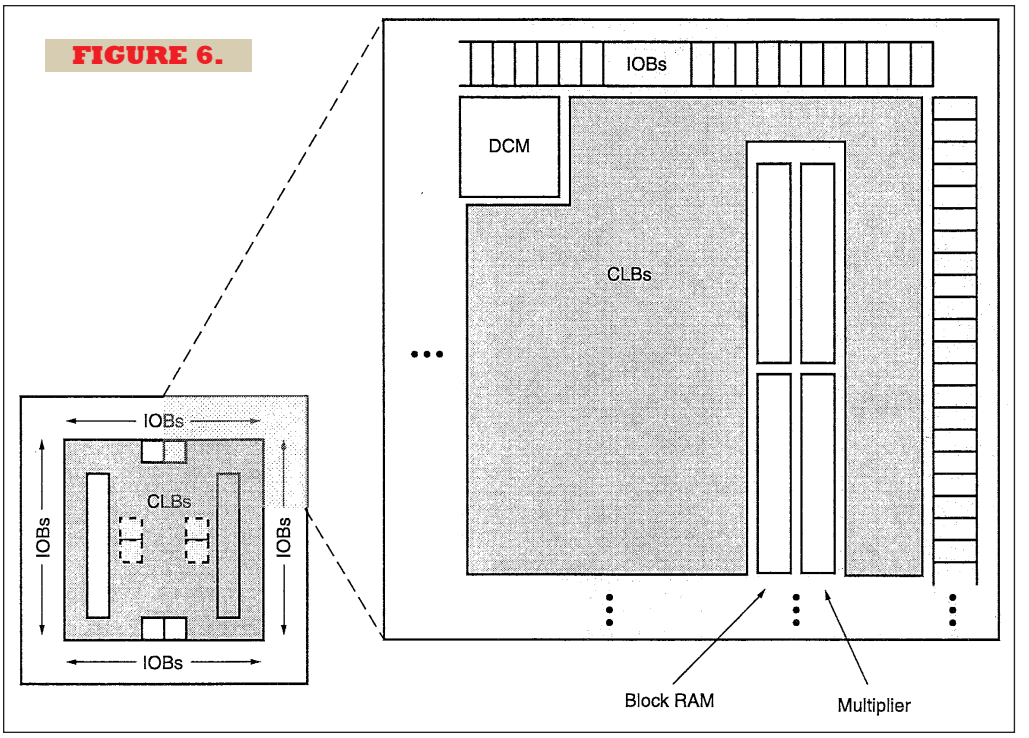


FIGURE 6.

that 22 x 16 = 352, not 240. This is because some of the CLBs are not there to make room for other structures which will be discussed in the next paragraph.

Also located in the FPGA are IOBs (input/output blocks). There are four IOB banks in the FPGA (see **Figure 5**). The IOB block takes care of the physical interconnections from the internal logic to the FPGA pins themselves. The IOBs can be configured to over 20 I/O standards and drive from 2 mA up to 16 mA. For the circuits that will be demonstrated in these articles, we will use the LVTTL (low voltage TTL) I/O standard where 3.3V is a logic "1." We can also set the drive at 16 mA to directly drive LEDs. The slew rate can be configured from slow to fast to match the appropriate devices the FPGA may be driving.

There are 72K bits of block RAM in the FPGA. There are also four 18-bit dedicated multipliers. There are two DCMs (digital clock managers) as shown in **Figure 6**. The block RAM, multipliers, and DCMs take up some of the CLB space; that is why there are only 240 CLBs instead of 352.

The XC3S100E on the Basys2 board is in a 132-pin package which allows for a total of 83 user I/O pins, although the PCB only makes 16 of these available at the four six-pin expansion connectors.

In order to generate the configuration bit files, Xilinx provides their ISE Design Suite webPACK as a free download; go to **www.xilinx.com**. The ISE Design Suite has a program called Impact which can be used to program CPLDs, FPGAs, and PROM chips, but it will not communicate with the Basys2 board. Digilent has a

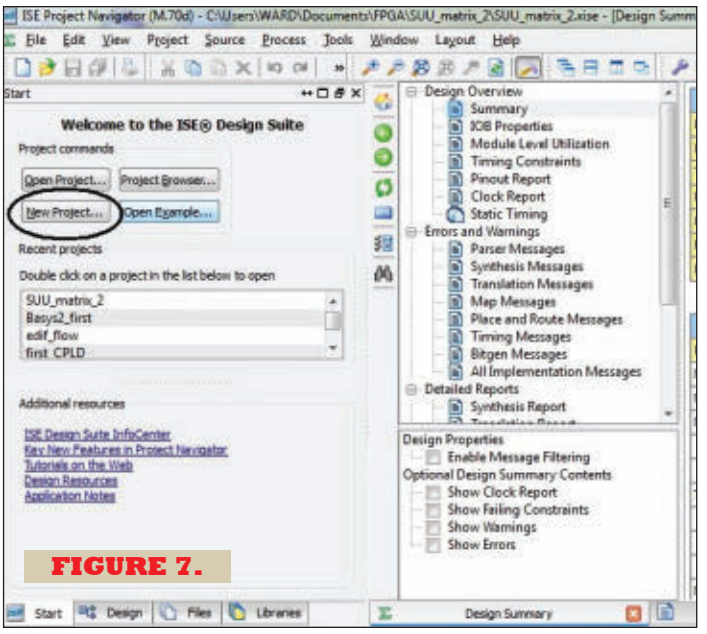


FIGURE 7.

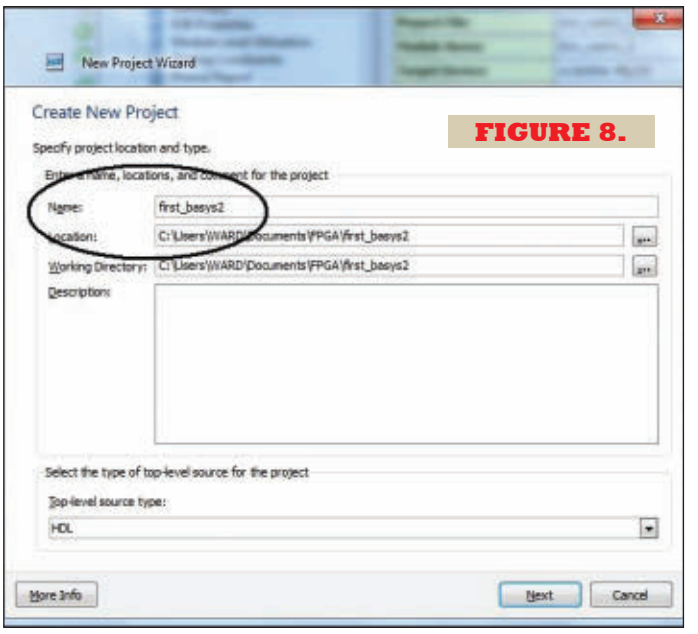


FIGURE 8.

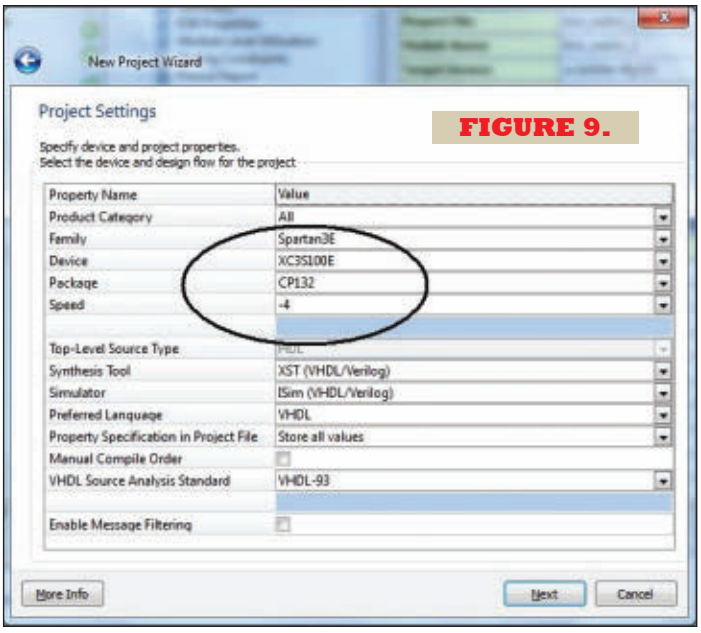


FIGURE 9.

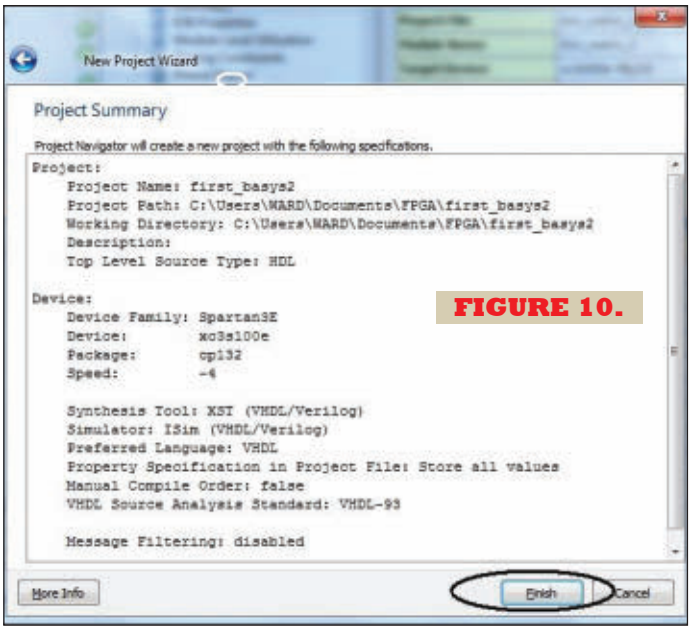


FIGURE 10.

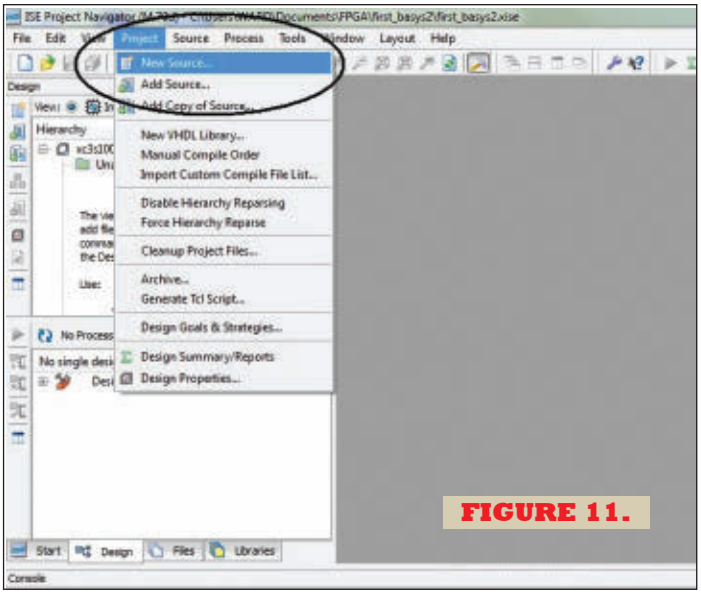


FIGURE 11.

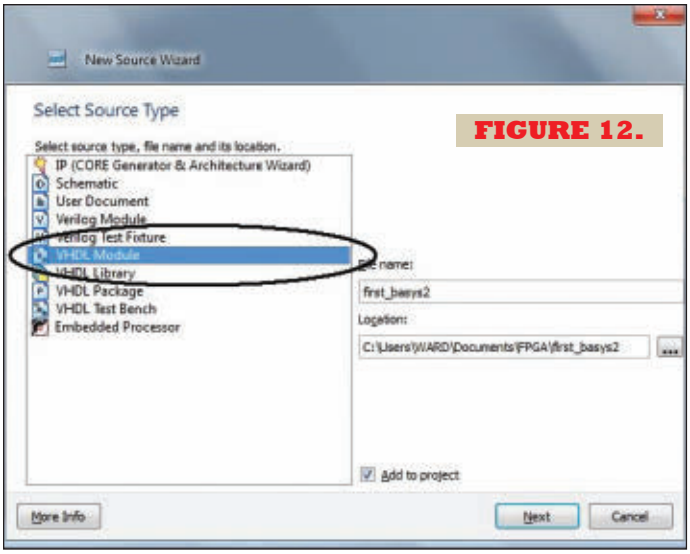


FIGURE 12.

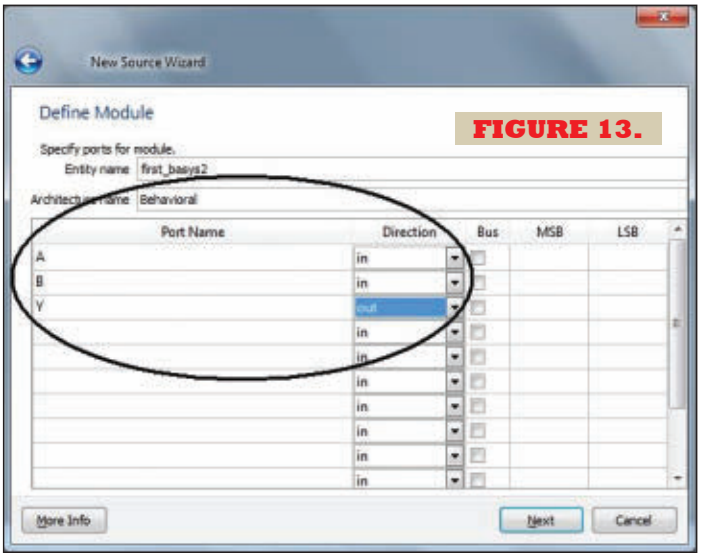


FIGURE 13.

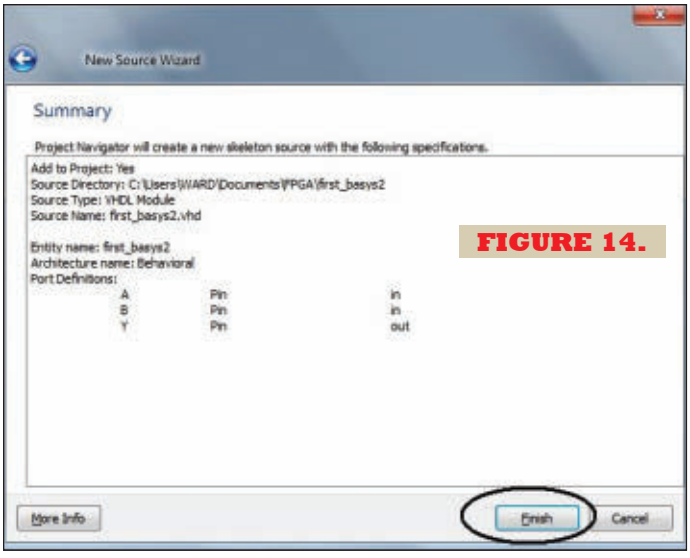


FIGURE 14.

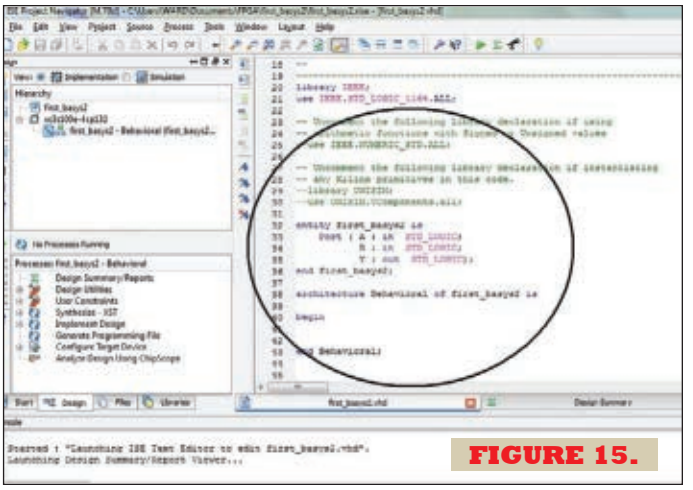


FIGURE 15.

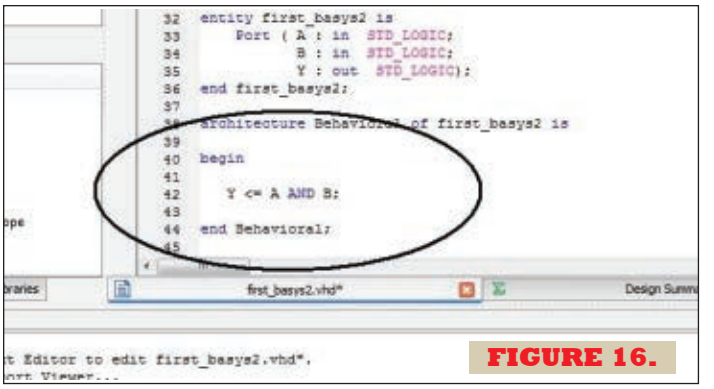


FIGURE 16.

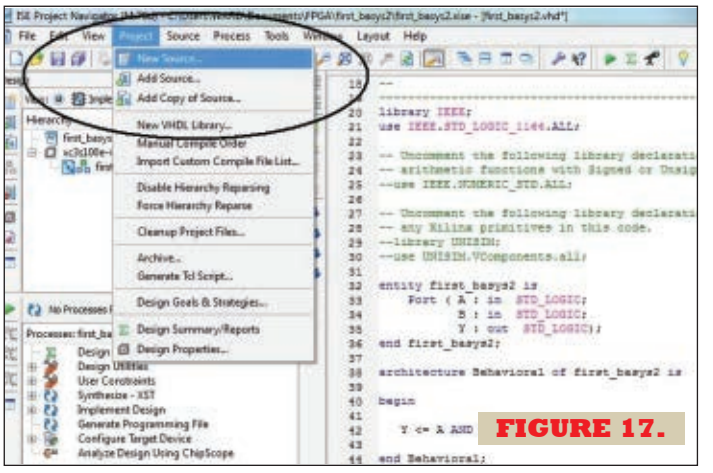


FIGURE 17.

program available that can be used to take the bit file compiled in the ISE Design Suite and program it into either the Basys2 FPGA or its PROM. Go to **www.digilentinc.com** and search for "digilent adept," then download and install the appropriate program for your computer. With both of these programs installed, you are now ready to build your first FPGA digital design.

Let's start out very simple and make a basic two-input AND gate using two of the Basys2 slide switches as inputs, and use one of their on-board LEDs as an indicator of the AND gate's output. This will be done in VHDL or very high speed integrated circuit hardware description language. Open the Design Suite and start a new project as shown in **Figure 7**. From the Create New Project window, type in a name and location for your project (see **Figure 8**). From the Project Settings window, select a Spartan3E, XC3S100E, CP132, -4 as shown in **Figure 9**. **Figure 10** shows the Project Summary window; click Finish to continue. Select Project and New Source as shown in **Figure 11**.

In the next window, click on VHDL Module and type in a file name and location for your source file (see **Figure 12**). **Figure 13** allows you to enter port names and directions; this can be done later but since this is a very simple circuit, it's easy to do it now. **Figure 14** is another Summary window; select Finish to continue on. **Figure 15** shows you the VHDL template that the ISE program created for you with your inputs (A and B) and your output (Y).

Most of the template is made up of comments; the green text preceded by two dashes (—) can be ignored or deleted. All we need to do is enter how we want these inputs and outputs to operate logically. Line 42 in **Figure 16** shows how to enter a two-input AND gate with A and B

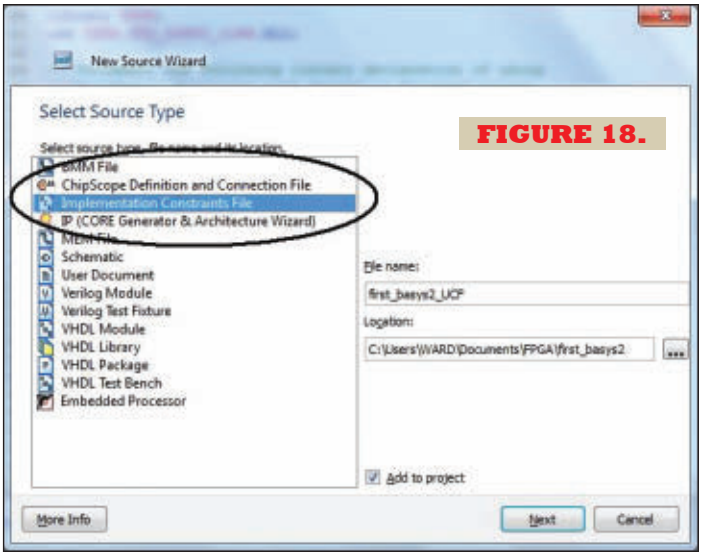


FIGURE 18.

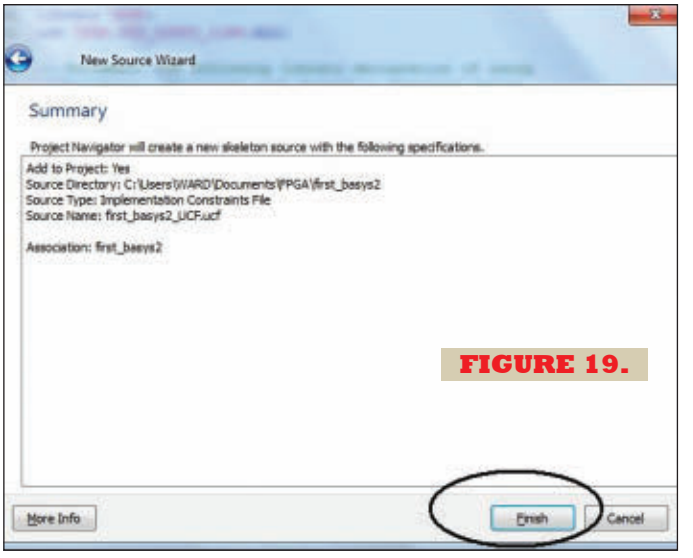


FIGURE 19.

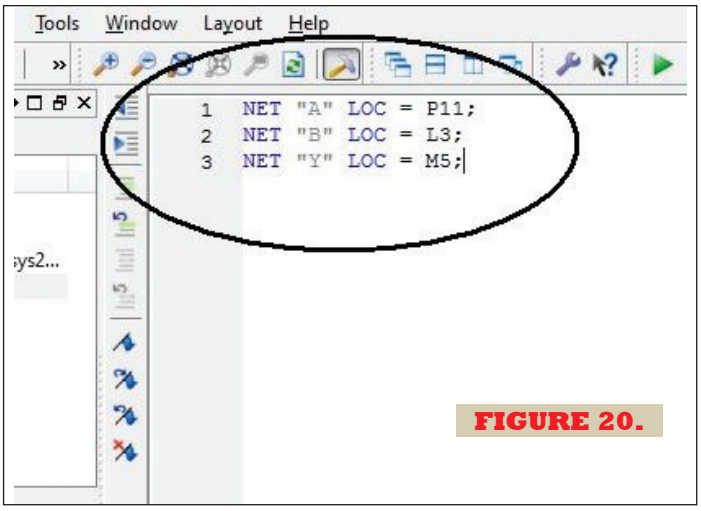


FIGURE 20.

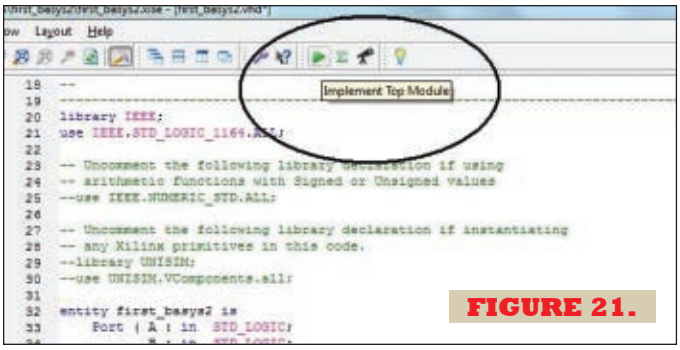


FIGURE 21.

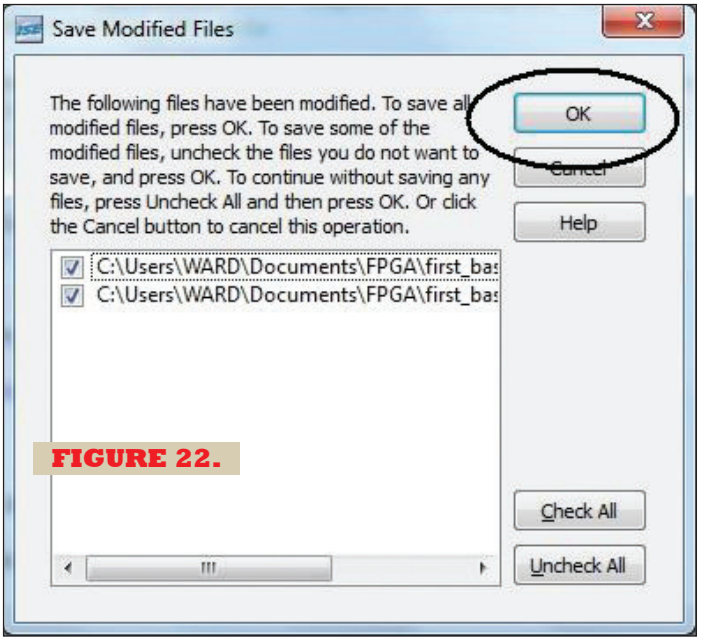


FIGURE 22.

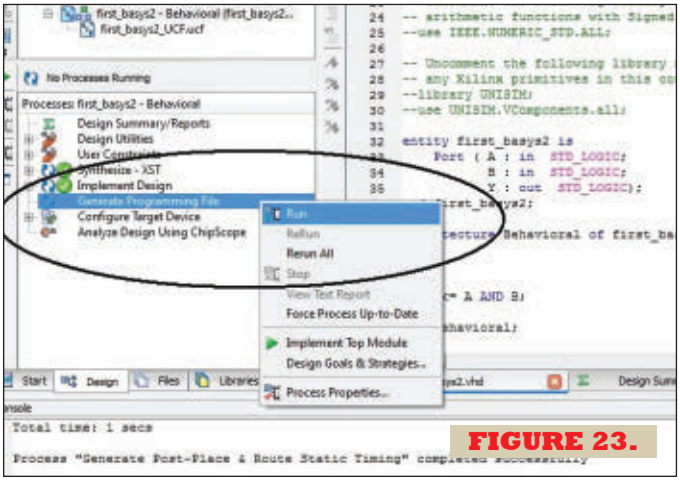


FIGURE 23.

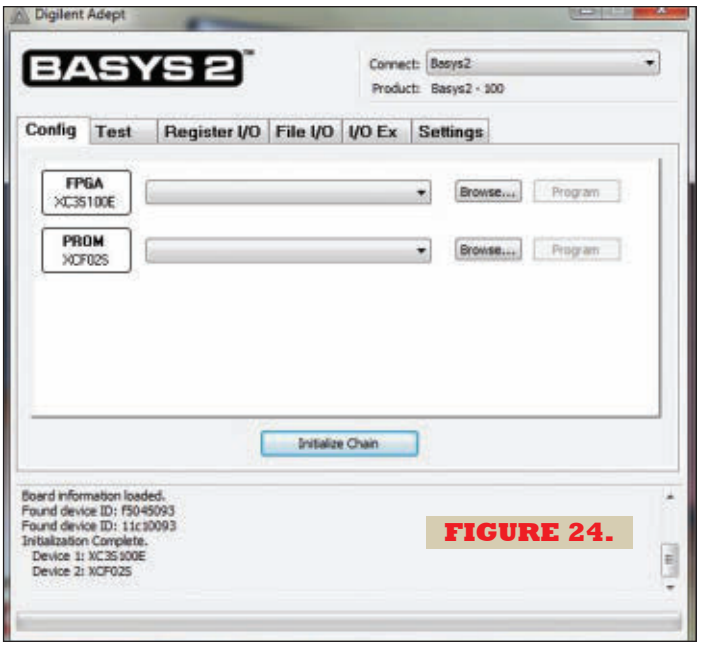


FIGURE 24.

as inputs and Y as the output. After line 42 is typed in, the VHDL listing could be compiled and fitted into the FPGA. However, since the Basys2 board has already connected certain FPGA pins to the slide switches and LEDs, we need to tell the program how we want these signals routed. This is done with a .UCF file or user constraint file.

To add a user constraint file, select Project New Source as shown in **Figure 17**. From the next window, click on Implementation Constraints File and type in a file name as shown in **Figure 18**. It's not a bad idea to add the letters UCF into the file name so it is easier to identify. Click Finish from the next Summary window as shown in **Figure 19**. You will now have a blank screen to enter your user constraint information.

All that is needed is to locate or "LOC" the appropriate net or port name with the desired FPGA pin number (see **Figure 20**). These FPGA pin numbers are printed on the Basys2 PCB next to each switch and LED. The information is also available in the Basys2 user guide which can be downloaded from the Digilent website. Now if the Implement Top Module icon is clicked on, it looks like a green RUN arrow. The project will compile and match your port names with the desired FPGA pins as described in the UCF file (see **Figure 21**). You may see the next window in **Figure 22** asking you to save these files; click on OK to continue.

If your original VHDL source file is okay (not missing semi-colons, etc.) and your UCF file is good as well, the program will work away for a few seconds. If there are any errors, you will see messages in the console window at the

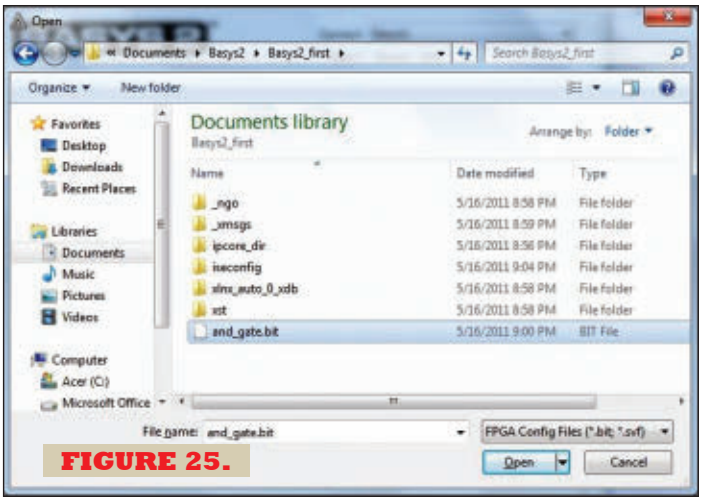


FIGURE 25.

bottom of the screen explaining what the problems are. If all goes well, you can now generate the necessary bit file that can be downloaded into the FPGA or PROM chip. Click on Generate Programming File and then Run, as shown in **Figure 23**. We now need to leave the Xilinx ISE Design Suite to get this bit file into the FPGA.

With your Basys2 unit plugged into a USB port and turned on, open the Digilent Adept program. The program should automatically recognize your Basys2 unit. At this point, it should be noted that there is a demonstration program already loaded into the PROM chip of the Basys2 board when you get it. It allows you to move the slide switches, turn on LEDs, and press the pushbuttons to turn off the seven-segment displays while they count up. If you load your AND gate into the PROM, you will overwrite this demonstration program and lose it. The program is available from the Basys2 web page if you want to download another copy; look for Basys2_100userdemocclk.bit.

If you load your AND gate bit file into the FPGA itself, it will run, but on powering down and up again the demo program will be loaded back into the FPGA. Click on the Browse button to the right of the FPGA drop down window (see **Figure 24**). Select the bit file that was generated previously from the Xilinx program as shown in **Figure 25**. Now you can click on the Program button. You'll probably get a clock error message a couple of times as shown in **Figure 26**. Select Yes; it works fine anyway. Now you can move slide switches SW0 and SW1 to their high positions and

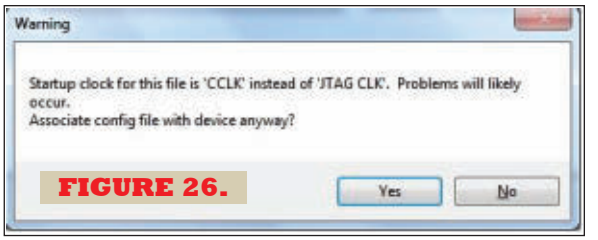


FIGURE 26.

LED0 should come on following the AND logic function that was placed into the FPGA.

We really haven't taken any time to explain VHDL during this article. If you want a good concise VHDL tutorial search for *The Low-Carb VHDL Tutorial* by Bryan Mealy on the Internet.

Next Time

The next article will demonstrate how to use the Basys2 FPGA system to control an 8 x 8 LED matrix to scroll a message across it. This will require an external breadboard and the 6" expansion cables. The VHDL code listing will also be explained in greater detail. Hopefully, this article has given you enough information to at least get you started experimenting with FPGAs. There are also several example programs for the Basys2 FPGA trainer available on the Digilent website. SV


EARN MORE MONEY
Get your dream job!
Be an FCC Licensed Wireless Technician!
Make up to $100,000 a year and more with NO college degree
Learn Wireless Communications and get your "FCC Commercial License" with our proven Home-Study Course!

• No need to quit your job or go to school.
• This course is easy, fast and low cost.
• No previous experience needed!
• Learn at home in your spare time!
Move to the front of the employment line in Radio-TV, Communications, Avionics, Radar, Maritime and more… even start your own business!
Call now for FREE info
800-877-8433 ext. 210
Or, online:
www.LicenseTraining.com
COMMAND PRODUCTIONS
Warren Weagant's FCC License Training
P.O. Box 3000, Dept. 210 • Sausalito, CA 94966
Please rush FREE info immediately!
NAME:
ADDRESS:
CITY/STATE/ZIP:
email: info@LicenseTraining.com

The NXT Big Thing #14

The Sound Of ... Robots?

By Greg Intermaggio

LDLDLDLDLDLLDDLLDLD! (Wookie for hello everyone!) In the last edition of The NXT Big Thing, we completed our gyro controlled robots. This month, we'll be tackling something a bit different: the sound sensor! Let's get started!

Understanding the Sensor

The sound sensor for NXT is a bit deceptive. Unlike a microphone, it can't identify specific sounds. Instead, the NXT sound sensor detects volume in decibels which means that its reading will always be the loudest sound it detects.

Today, we'll be using the sound sensor to make Eddie do a few different things — from running from sound, to steering based on how loud you are, to drawing an awesome sound gauge.

The Speed of Sound

We'll start by making Eddie react in his movement to the sounds he hears. If Eddie hears a loud noise, he'll move very quickly; if he hears no noise, he'll stop.

First, we'll need to build the sound sensor.

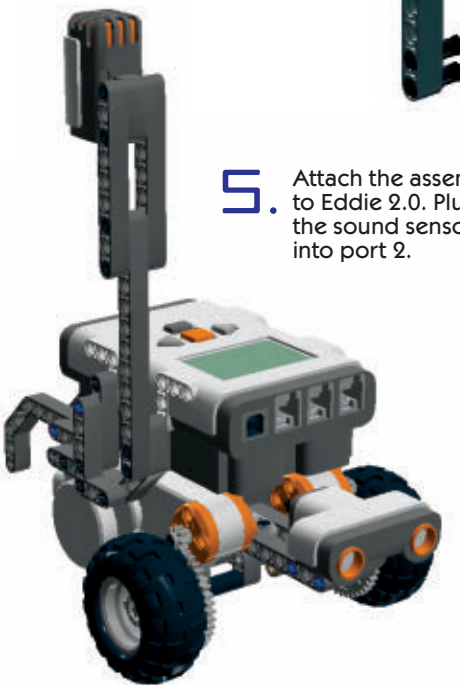
Building Instructions: Sound Sensor Attachment

1. Start with a 13-hole studless beam. Snap in two standard friction pins and two double friction pins as indicated.

2. Add a nine-hole studless beam with two double friction pins.

3. Attach the sound sensor.

4. Close the assembly off with another nine-hole studless beam.



5. Attach the assembly to Eddie 2.0. Plug the sound sensor into port 2.

attachment for Eddie.

Okay, now that the sound sensor attachment is built and attached, let's get to the program (see below)!

Go ahead and test it out. If you make a loud noise, Eddie should move quickly away; if you make less noise, he'll slow down.

A Noise Controlled Robot?

So, what else can we do with that sound sensor? Well, if we change up our program just a tad, we can get Eddie to determine his direction based on the volume of the noise he hears. Let's give it a go!

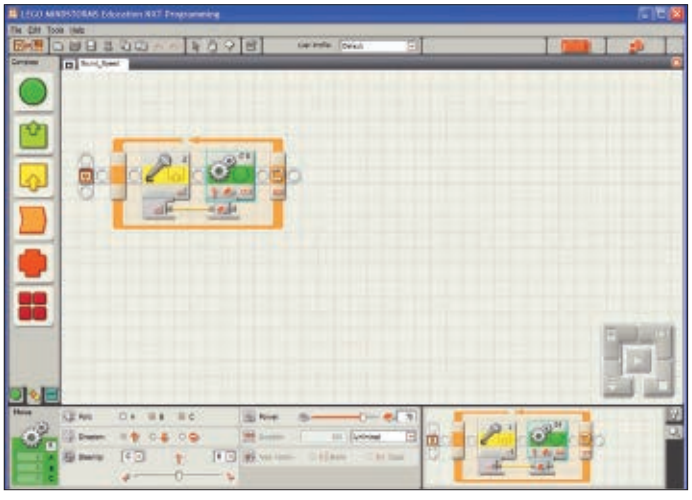
The sound sensor will give a value between 0 and 100. A value of 0 means silence and a value of 100 means loud noises! Recall that we have a data port on the motor to control steering dynamically. This port interprets a value of -100 as a hard left turn; 0 as straight forward; and 100 as a hard right turn.

If we wanted Eddie to be very precise in how he reacts to sounds, we'd add a step and make a sound value of 0 equal a steering value of -100 and a sound value of 100 equal a steering value of 100. That said, our goal here is to make Eddie easy to control by sound, so we'll make his turns more gradual by making the harshest steering values -50 and 50 instead of -100 and 100.

Give it a go. If you're really quiet, Eddie should turn left. If you're really loud, Eddie will turn right, and if you're just right, he'll go straight!

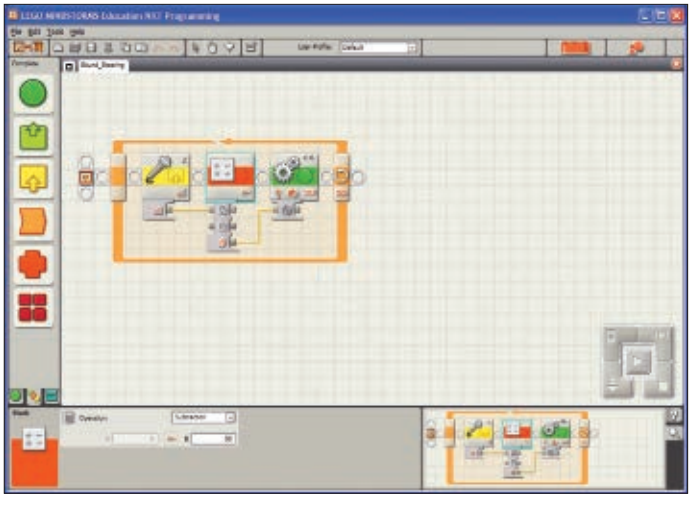
A Bigger Challenge

Okay. Now that we've done some experimenting, let's make something awesome! We'll use the sound sensor to



Sound Speed Program Instructions

Figure 1.
- Create a program called Sound_Speed.
- Drag in a loop, inside of which your program will take place.
- Add a sound sensor block and make sure it's set to port 2.
- Add a motor block and set motors B and C to forward.
- Run a data wire from the output of the sound sensor block to the power data port of the motor block.



Sound Steering Program Instructions

Figure 1.
- Bring up your Sound_Speed program, and save it as Sound_Steering.
- Add a math block and wire the output of the sound sensor block to A. For B, choose 50; for operation, choose subtraction.
- Wire the output of the math block to the steering data port of the motor block.

create our very own custom sound gauge.

Test it out! Eddie should show a bar on his screen based on how much sound he hears. If your sound bar looks crooked, you probably forgot to set the coordinate values to 0 in Step 3.

Challenge Ideas

Now that you understand the sound sensor a bit better, here are some fun programming challenge ideas:

- Make a vertical sound gauge.
- Program a sound monitor that looks like an actual sound recorder program, recording volume levels over time.
- Use two sound sensors and some sort of insulating material to make their hearing directional to create a robot that can follow (or avoid) sound.

Sound Gauge Program Instructions

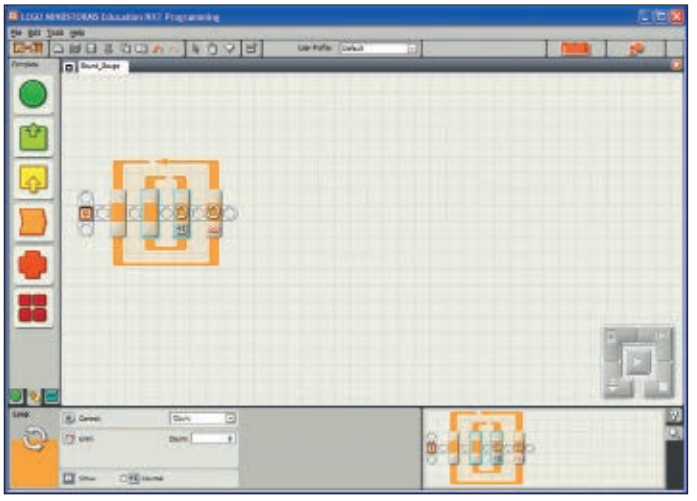


Figure 1. Create a program called Sound_Gauge. Start with a loop inside a loop. On the inner loop, set the Control to Count, and the Count to 5. This means that the inner loop will run five times per cycle. We're going to use one loop per pixel width of our sound gauge; you can adjust this value later.

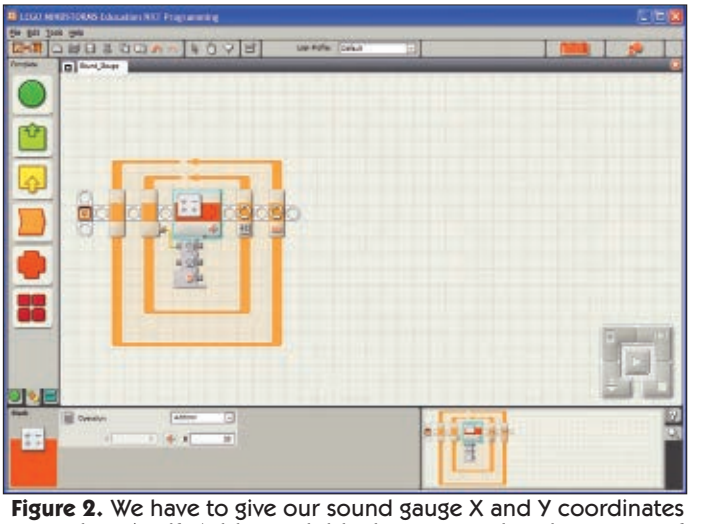


Figure 2. We have to give our sound gauge X and Y coordinates to draw itself. Add a math block connected to the output of the loop, and set the operation to addition and B to 20. This will ultimately tell Eddie what Y coordinates to draw from. Now we just need X.

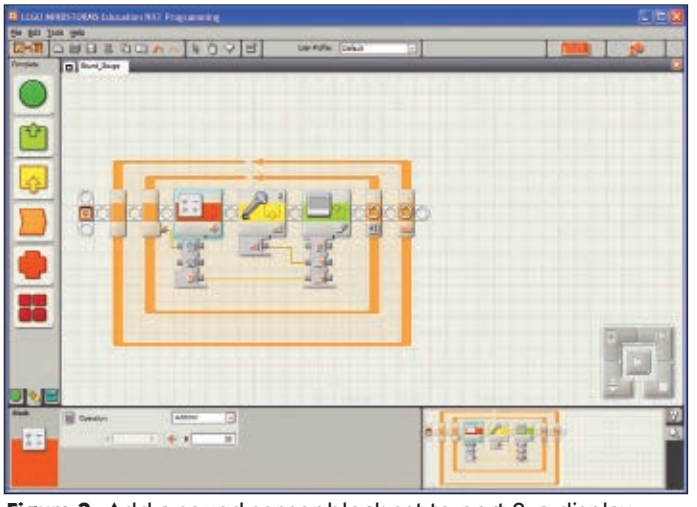


Figure 3. Add a sound sensor block set to port 2, a display block with Action set to Drawing, and Type set to Line. Set all the coordinates to 0, and expand the data hub.

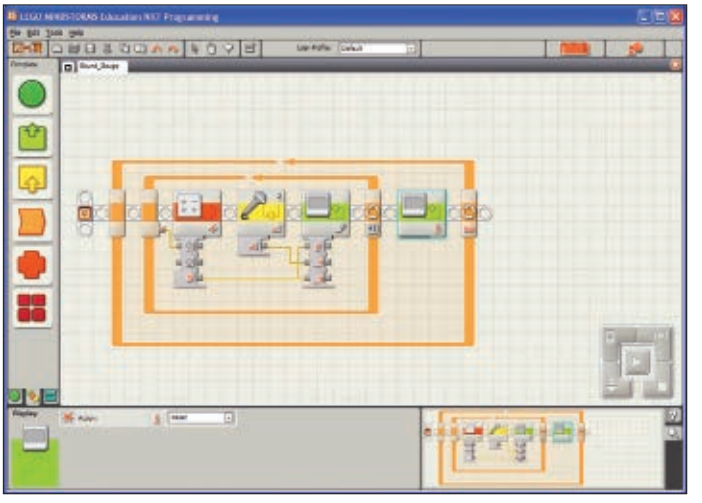


Figure 4. Run a data wire from the output of the math block to the end point Y data port, and another data wire from the output of the sound sensor block to the input of the end point X data port.

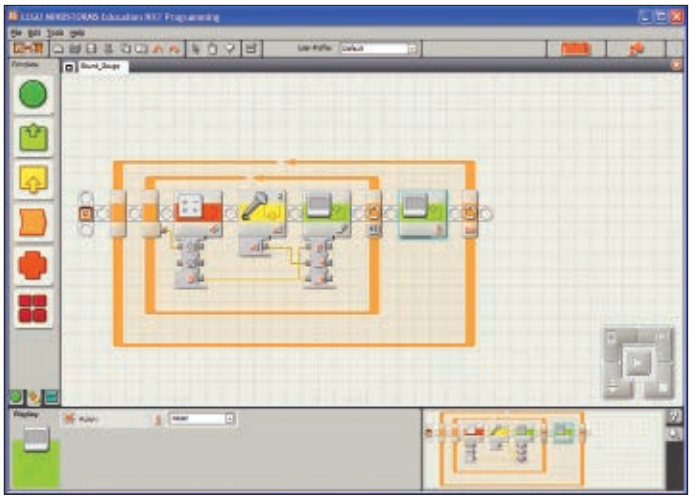


Figure 5. Add another data wire from the math block; this time to the Y data port. Also, add a display block as shown — outside of the inner loop and with Action set to Reset. This will reset the screen after each time Eddie generates his sound gauge, so that it doesn't overlap itself.

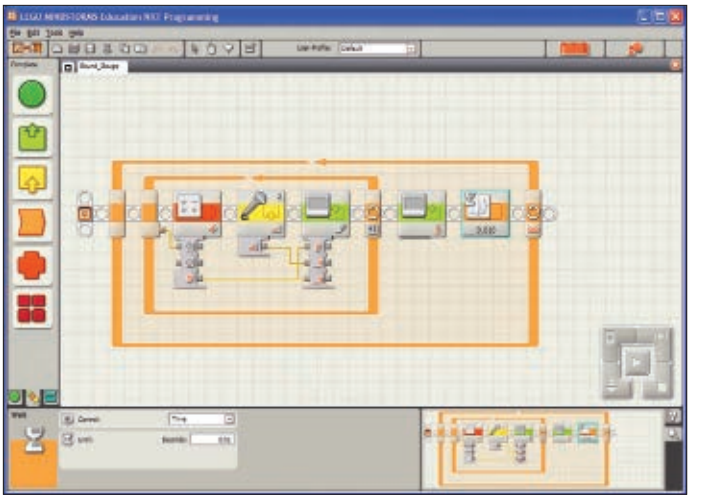


Figure 6. Finally, add a .10 second delay. This will make Eddie generate the sound gauge 10 times a second.

Wrapping Up

Hoorah! We've just delved into some awesome programming projects with the sound sensor. We learned how the sensor works, plus three different cool ways to use it!

Be sure and join us again next month for The NXT Big Thing! **SV**

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

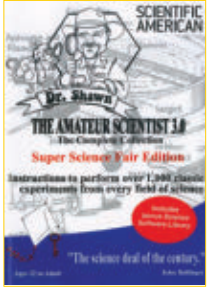
ROBOTICS

PIC Robotics
by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.
$24.95

The Amateur Scientist 4.0 The Complete Collection
by Bright Science, LLC



There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!
Reg $26.95 Sale Price $23.95

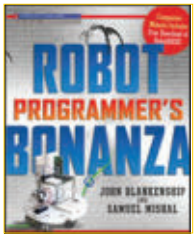
Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists
by Dustyn Roberts

In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.
$29.95*

Build Your Own Humanoid Robots
by Karl Williams
GREAT 'DROIDS, INDEED!

This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.
$24.95*

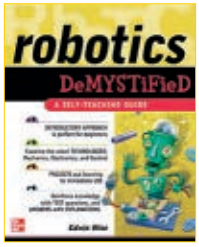
Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.
$29.95

Robotics Demystified
by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

Published by T & L Publications, Inc.

$57 per bundle

Save $10 off the normal price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Call my Webstore and you'll get someone in AMERICA!

Visit my online store @ www.servomagazine.com

LEGO MINDSTORMS NXT Idea Book

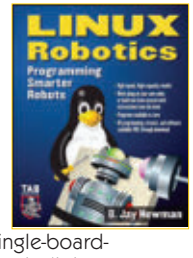
by the Contributors to The NXT Step Blog

If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog – the authoritative online source for MINDSTORMS® NXT information and advice – has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

Reg $29.95 Sale Price $24.95

Linux Robotics

by D. Jay Newman



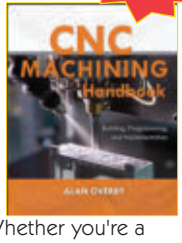
If you want your robot to have more brains than microcontrollers can deliver – if you want a truly intelligent, high-capability robot – everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

$34.95

CNC Machining Handbook: Building, Programming, and Implementation

by Alan Overby

NEW!



The *CNC Machining Handbook* describes the steps involved in building a CNC machine and successfully implementing it in a real world application. Helpful photos and illustrations are featured throughout. Whether you're a student, hobbyist, or business owner looking to move from a manual manufacturing process to the accuracy and repeatability of what CNC has to offer, you'll benefit from the in-depth information in this comprehensive resource. **$34.95**

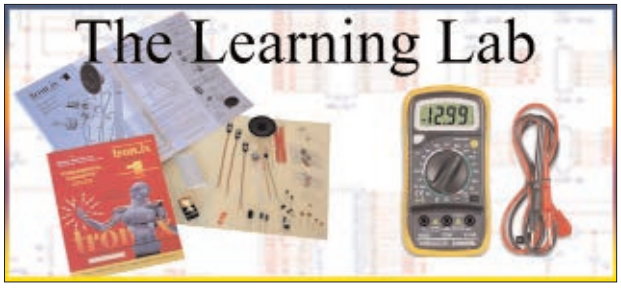
SPECIAL OFFERS

RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal

If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price $14.95 Sale Price $9.95



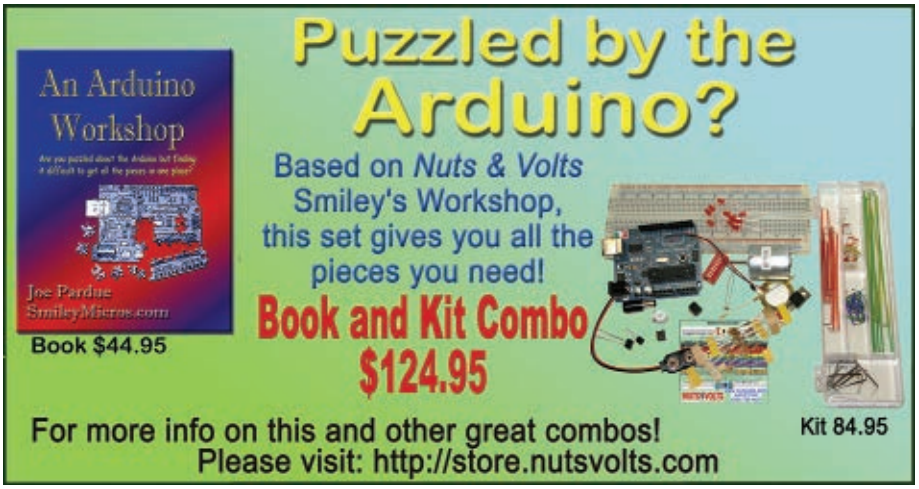
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

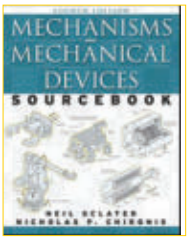
Regular Price $79.95 **Subscriber's Price $75.95**

Or order online
www.servomagazine.com

SPECIAL OFFERS



Mechanisms and Mechanical Devices Sourcebook

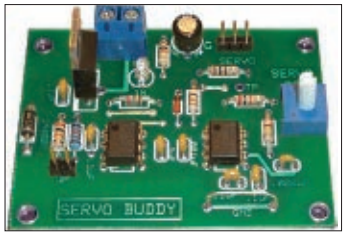


by Neil Sclater, Nicholas Chironis

Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg $89.95 Sale Price $69.95**

PROJECTS

The SERVO Buddy Kit

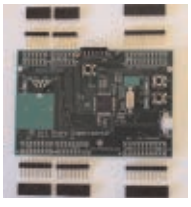


An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

32-Bit Micro Experimenter Board

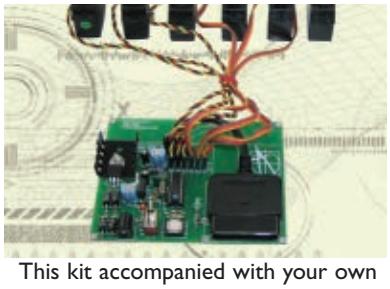


The new 32-Bit Micro Experimenter is the fastest way to learn 32-bit microcontrollers.

NEW!

The kit includes onboard 46 programmable I/O and USB, free software, carefully documented step-by-step experiments for USB, embedded web server, graphics and audio, wireless, RTOS, and file I/O. User pushbuttons, LEDs, and 32 kHz clock crystal. Can be used in solderless breadboard environment or stand-alone. Also supports Arduino compatible interface.
Subscriber's Price **$89.95**
Non-Subscriber's Price **$93.95**

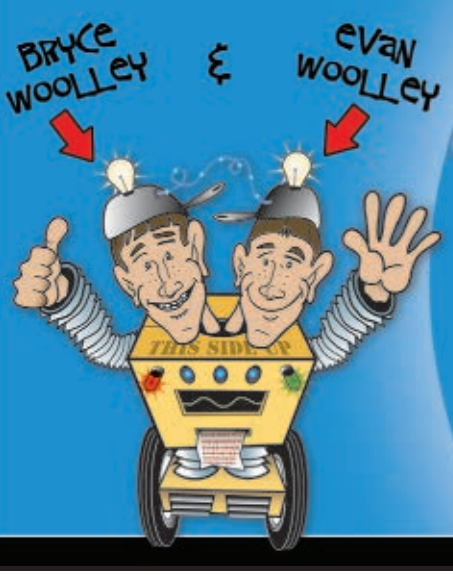
PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*.
Assembled units available!
Subscriber's Price
$79.95
Non-Subscriber's Price
$84.95

Twin Tweaks



THIS MONTH:
Getting Serial

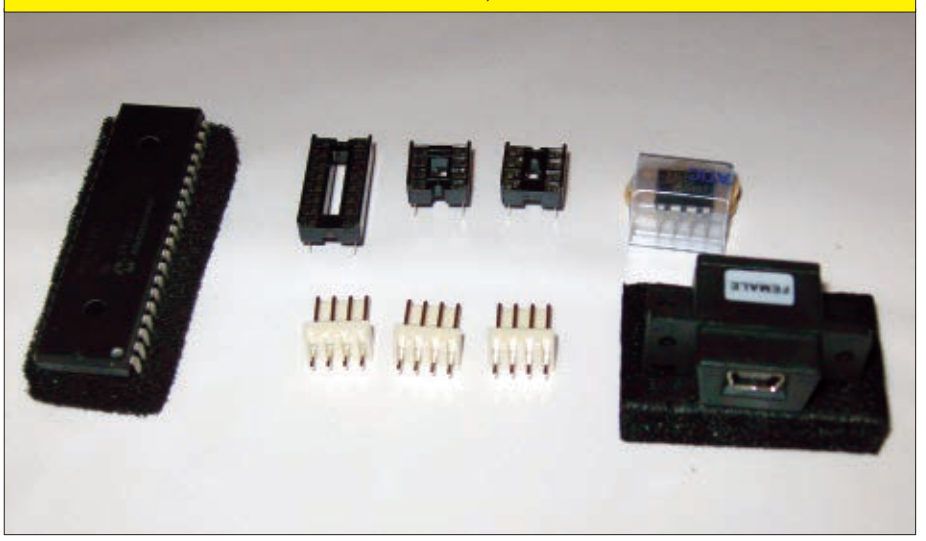
THE MARK III ROBOT FROM PARTS.

Even casual readers of *SERVO* have probably picked up on one of our major pet peeves when it comes to robotics kits: RS-232 serial ports. We can tolerate a little nostalgia as well as anyone, but the frequency with which we see RS-232 ports on robot kits is baffling. Robotics tinkerers are generally high tech people with high tech gadgets, and we can't remember the last time we saw a relatively new laptop with an RS-232 port on it. Roboticists like us are forced to invest in a serial-to-USB adapter which may not be a big expense, but it certainly seems like an unnecessary inconvenience.

Back in the July '09 issue of *SERVO,* we dedicated an article to unraveling the mystery of RS-232 port persistence. As part of the article, we wanted to convert one of our numerous serial equipped kits with a USB connection. Our plans were foiled when we received the wrong part in the mail, so instead we took a more academic approach — even interviewing one of Evan's professors from UCSD. The results of our investigation were clear: There was really no good reason for robotics kits to cling to serial ports like dust to a Swiffer. Even so, we've still been haunted by our unfinished business of actually converting a robot kit from a serial to USB connection.

Thus, we arrive at the present article. *SERVO* readers may also be acquainted with the Mark III robot kit from Junun Robotics, as it has made several appearances in Twin Tweaks over the last few months. We found an FTDI adapter through Digi-Key that we could use to convert the RS-232 equipped bot to a much more current USB connection. With this part on the way, we could finally answer the last remaining question in our crusade against RS-232 ports: For tinkerers fed up with using adapters, is converting a robotics kit to a USB connection a practical endeavor?

CONNECTORS AND MORE, FOR MODS GALORE!



Mark My Words

The Mark III is a compact little robot kit that uses a PIC microcontroller as its main brain. As its name suggests, the general-purpose robot evolved from two previous incarnations — both of

which specialized as mini Sumo robots. The Mark I was originally developed by Marvin Green of the Portland Area Robotics Society (PARTS). The Mark I was meant to serve as an easy-to-use robotics kit that aspiring roboticists could use in the first annual PARTS MiniSumo competition held back in 2000. Improvements were made and the Mark II was born for the second annual competition in 2001. The Mark II included upgrades like infrared sensors (the classic Sharp GP2D12 sensors) to detect the opposing robot. The Mark II eventually evolved into the current Mark III, with the main goals of making the robot general-purpose and expandable.

The CPU for the Mark III is a PIC16F877 from Microchip. The Mark III is OOPic compatible and, in addition to this programming flexibility, it is also amenable to a number of physical embellishments called mezzanine boards. The mezzanine boards can mount on top of the controller board and delightfully expand the capabilities of the kit. The mezzanine boards range from a breadboard-like prototyping platform to a sensor board equipped with servo headers and a buzzer, among other bells and whistles.

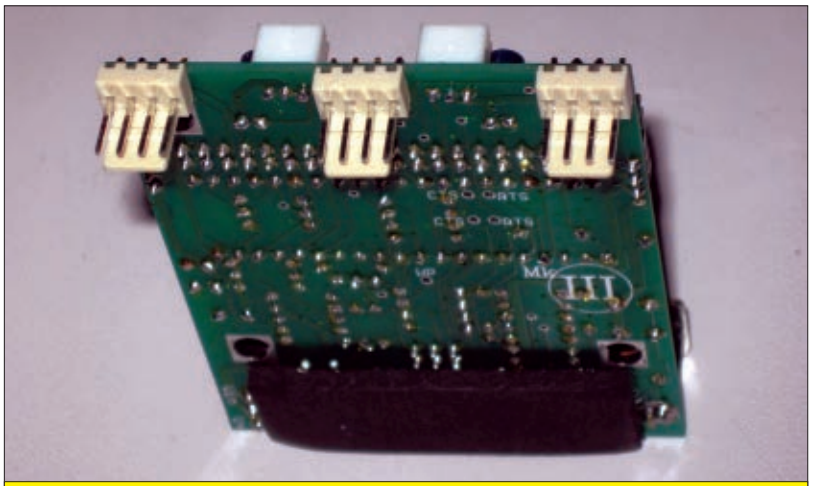
On the mechanical side, the Mark III is driven by two GWS servo motors, and it rides atop two plastic injection molded wheels. The Mark III retains the compact frame of its mini Sumo pedigree — complete with the front wedge and low ground clearance.

A Robot For All Seasons

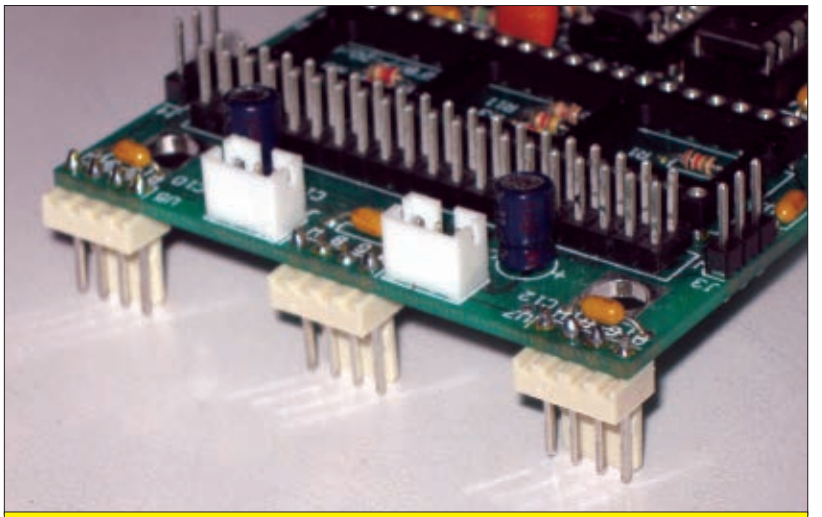
As much as anything else, this is a tale about the evolution of a robot. We admire the Mark III for being the product of an independent robotics group, and we think the real challenge of a product like the Mark III is to keep up in a market saturated by the next version of whatever you just bought, and where everything moves on Apple time (which must be the polar opposite of geologic time). Our experience with the Mark III began around 2007, when our dad ordered the kit as a fun project to work with in his spare time, and in particular as a way to begin working with OOPic.

For starters, the Mark III board must be soldered together from scratch. While some might find this endeavor tedious, we always found the step-by-step soldering of circuits to be a languidly pleasant affair and a great way to become acquainted with the kit. The Mark III PCB is meticulously labeled, and putting everything in its proper place is a very straightforward task.

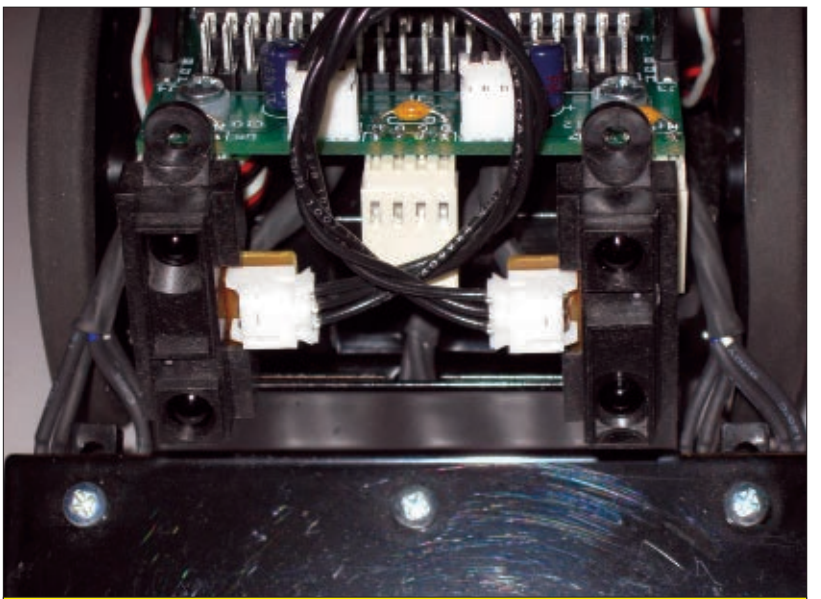
Even this initial assembly presented some opportunities for improvement. We've always



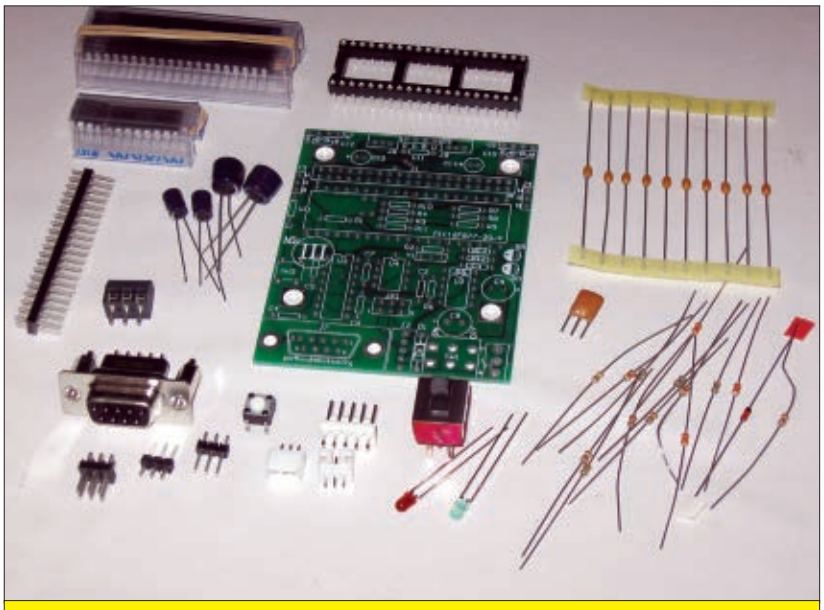
The Mark III board outfitted with connectors for the infrared sensors.



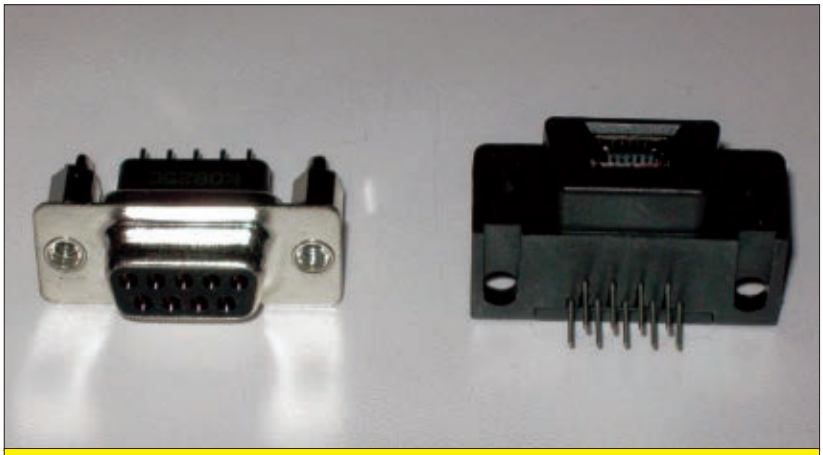
Keeping things clean with connectors.



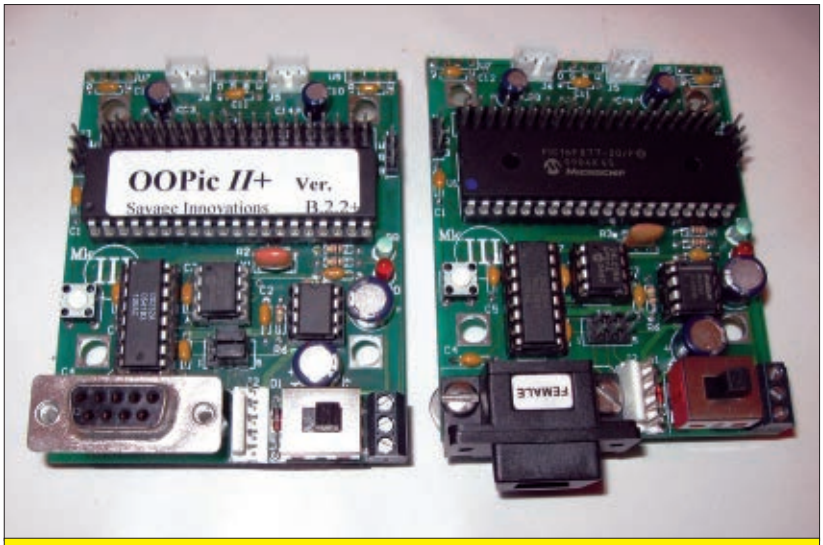
Wiring up the infrared sensors.



Starting from scratch with a new board.



Comparing the serial DB9 connector and the FTDI adapter.



We switched from RS-232 to USB and saved a ton of hassle with adapters!

been wary about soldering ICs directly to PCBs — desoldering every pin from a dual inline package can be a real test of patience if you ever want to make a change. In case we ever wanted to switch out any of the components, we used sockets for the DIPs that were soldered to the board instead of the ICs themselves. The sockets were still low profile enough to allow the addition of mezzanine boards without incident.

Our socket mania reached more than just the ICs. We also added sockets for the Sharp sensors which would normally solder directly into the front of the PCB. This would make things easier if we ever wanted to build a new board for the Mark III — no desoldering, just unsocketing. The additional sockets were pretty low profile and did not interfere with the compact frame of the robot. The sockets also made it easier to route the wires for the sensors while avoiding some of the stress relief problems that attend direct joints with the PCB.

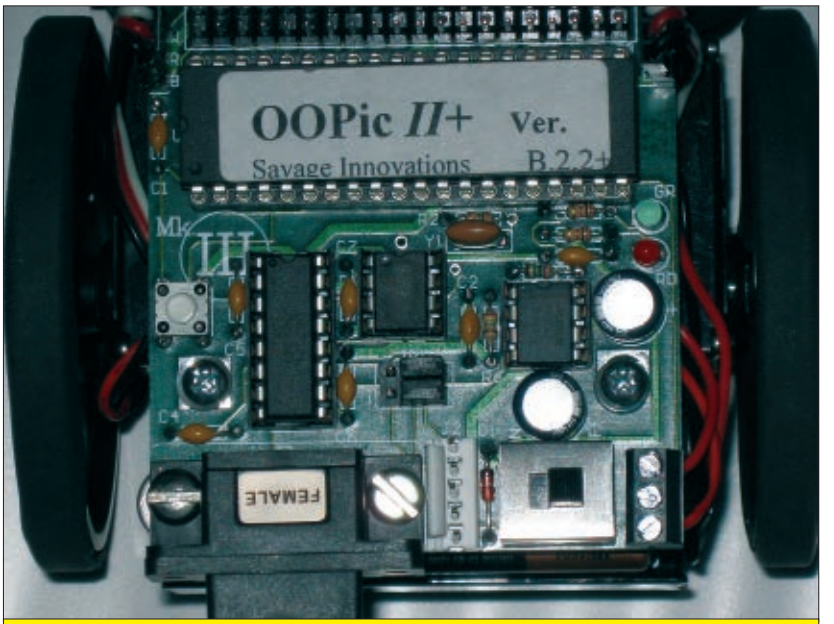
As mentioned previously, one of the main reasons our dad was initially drawn to the Mark III as an exciting platform for experimentation was the prospect of using it as a way to become familiar with OOPic. The Mark III is compatible with an OOPic microcontroller that could effortlessly replace the original PIC because they both fit into the socket we used while originally building up the board. The Mark III is a great way to experiment with object-oriented programming because it comes with sensors (like the Sharp infrared sensors) that can be turned to classic tasks like line following.

The dual servo motor drive is also ideal for simple dead-reckoning programs. The prototyping and sensor mezzanine boards are a great way for robotics experimenters with limited time to tackle discrete projects that still reward the successful tinkerer with a sense of accomplishment. You could add a bumper style touch sensor so that the bot could wall-crawl around a maze Roomba style. You could add more infrared sensors to the front of the robot to improve its line following or opponent finding ability. Discrete projects like this are perfect for folks like our dad — someone who hasn't had as much time to focus on robotics projects since FIRST Team 1079 stopped pulling all-nighters in Robot Central (check out the September through December '04 issues for more on those glory days of yore).

Even with only limited time to devote to the Mark III, our dad has turned the bot into a scrappy competitor that has made several appearances in Twin Tweaks over the years. Whether it's smoking the Ollo bugs in a lopsided

drag race (see the April '09 issue for more) or tipping over glittery ramps in the Cirque du Robot (January '11), we have enjoyed using the Mark III as a sometimes hapless foil for our featured projects.

Most of the tasks we have enlisted it for have played to its strengths like line following around a track. Other times, we have deliberately taken advantage of its Sumo inspired low ground clearance, and in the Cirque du Robot we were still surprised at how high and steep we had to make our ramp obstacles so that they would tip over the low-riding bot. No matter whether we used the Mark III as a ringer, a tomato can, or something in between, we think it is impressive in the sense that it has remained a viable competitor in featured projects over the years.



THE BOARD OUTFITTED WITH THE FTDI ADAPTER.

The Solution To The Final Problem

The inspiration for this upgrade to the Mark III kit actually came from *SERVO Magazine* itself. The FTDI adapter necessary to convert an RS-232 kit to USB was featured in the August '10 issue in the Robytes section. The mod USB RS-232 embedded retro DB9F — as it was officially designated in the Digi-Key catalog — looked like the perfect solution to the persistence of the RS-232 ports. It was compact, soldered directly to the PCB through holes, and it was affordable. A single unit costs about $22. The FTDI adapter was designed specifically to upgrade devices still relying on the quaint DB9 connector, and this explains its somewhat bulky appearance when compared to USB ports we all know and love. The FTDI adapter was designed to avoid compatibility problems with the existing PCB, and so it mimics the footprint of the DB9 connector.

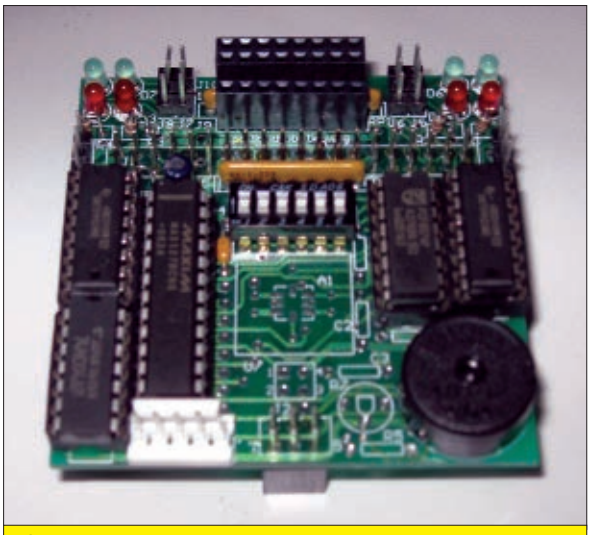
On the technical side, the FTDI adapter includes the FTDI FT232R USB-serial bridge IC within the unit, along with the requisite level shifters, so implementing the adapter is really a matter of plug and play. There are a few historical caveats, however, and they are candidly laid out in the datasheet provided by FTDI for the adapter. The familiar DB9 pinouts are the result of the PC-AT standard introduced by IBM in the early 1980s. Not every single device adopted the standard, though after this many years it is exceedingly unlikely that anyone using the adapter will be foiled by nonstandard pinouts. The FTDI adapter uses the standard pinouts, as we expect would be the case with whatever robot kit you feel like bringing into the modern age.

It would have been easy enough to desolder the DB9 connecter and replace it with the FTDI adapter, but we wanted to build up a new board anyway and this would allow for more visually stunning side-by-side comparisons. The FTDI adapter did indeed retain the footprint of the DB9 connector, and thus avoided any problems with running into the ICs and capacitors that crowded the PCB.

The only other step required to implement the adapter was to download the drivers from FTDI which can be achieved by a quick visit to **www.ftdichip.com**. The drivers downloaded without incident, and we hoped that actually programming the robot would not present any mysterious problems. We've all been there – you've painstakingly built up your robot, making sure every resistor has the right color code and every diode is pointed in the right direction, only to have a cryptic error message display when you try to download a program to the robot. To us, this can be one of the most vexing stages of any project, perhaps because

FREE FROM RS-232 AT LAST!

Programming the Mark III via USB.



An expansion board for the next evolution.

beyond checking the designation of the COM port by the programming software, further troubleshooting can be a murky affair. Also, with his mechanical engineering background, Evan is generally inclined to chalk up unforeseen problems to the software side.

Thankfully, the immortal conflict between the mechanical and software sides of a project did not play out in this case, because the FTDI adapter worked just fine. We were able to download programs just as seamlessly as before, but without the middleman of our Keyspan adapter. If anything, the end result of the project was almost a little anticlimactic – no last minute troubleshooting, no epic struggle against the knee jerk resistance to new USB technology. Everything went according to plan, and we were back to perfecting the Mark III's line following program in no time.

Evolutionary Robotology

We had finally done it. We finished the journey upon which we were side-tracked by a parts mix-up years ago. We had conclusively proven that switching over a robotics kit from an RS-232 connection to USB was no big deal. Or, was it? The transition was simple and seamless – it only took one part, as much soldering as you would be doing anyway, and no headaches when trying to reconnect the robot to the computer. Should this be ground shaking? It demonstrates that there really is no good reason for robotics kits to still come with RS-232 ports!

There were a lot of things that could have made the conversion from RS-232 to USB less than ideal. Perhaps our biggest concern was not the functional compatibility of the new adapter regarding the actual communication between the robot and the computer, but rather the physical compatibility with the robot board and the adapter. The Mark III board — as is to be expected from a robot with a strong mini Sumo pedigree — is very compact and doesn't necessarily have the real estate to devote to a bulky new connector. The FTDI folks had anticipated that very concern, however, by mimicking the footprint of a DB9 connector.

Other potential snags included problems with the device drivers or (on a more basic level) the design of the adapter could have been much more of a burden for the user to implement. The sleek casing of the unit encompasses two major components: the USB-serial bridge IC and the level shifters. Perhaps if those components could only be acquired separately, it might be more of a chore to implement them within the footprint of the DB9 connector on a crowded PCB.

All of these things could have happened and might give a semblance of a good reason as to why RS-232 ports persist on so many robotics kits. However, none of them did. The FTDI adapter is a simple-to-implement upgrade that will let you ditch the middleman serial-to-USB cord in no time. If only the folks that made the kits would take that step instead of having the end users do it ...

The FTDI adapter itself will run you about $20, so from a cost standpoint it is not necessarily a big advantage (or disadvantage) compared to buying an adapter cord like the Keyspan adapter that we're fond of. That said, comparing the cost to the user of modifying their own kit with the cost to a company of using USB ports in the first place is not an apples-to-apples affair. Kit producers have to include some kind of port for connecting with the computer, and we can't see how there would be too much of a difference cost-wise between the DB9 and USB Micro-B — particularly when larger scale producers get the benefit of price breaks associated with buying in bulk.

The great irony of this project is that out of all of the kits that we have worked on over the years, the Mark III's retention of the RS-232 port is probably the least offensive. The Mark III was developed by an independent robotics group in the early aughts, back when using a DB9 connector

was reasonable because laptops still had DB9s themselves. It does appear that something like the FTDI adapter might be somewhat more expensive for a small producer, and this might have wreaked havoc on the group's price point. The Mark III is available for only $92, and we can understand how some modifications might take the kit out of the preferred double digit territory. That irony, however, is also a victory for the Mark III.

The moral of this story is that good robotics kits — to stay relevant in a world that upgrades at the speed of Moore's Law — need to evolve and adapt. One way to do that is to demonstrate that leaving behind the vestiges of old and increasingly outmoded technology need not be an expensive, difficult, or unsightly affair.

Making Its Mark

Even though the Mark III was originally developed years ago, and even though we had been slowly making modifications to the kit since 2007, the tenacious robot has remained an exciting platform and a worthy opponent even for the newest kits. We are pleased to report that some of those newest kits that we've worked on recently (like the Robot Shop Rover and the upcoming USB I/O Explorer from Digilent) actually come with USB ports. For all of those folks with beloved kits that they have worked on over the years or new kits that inexplicably retain an RS-232 port, the process of upgrading your robot to modern-day USB technology is a surprisingly affordable and painless process. With one simple adapter from the appropriately named Future Technology Devices International (FTDI), you too can bring your robot into the future and keep on tinkering.

As for the Mark III, we recently assembled the sensor board expansion kit. The mezzanine board was only $12, and will allow for the addition of servos, switches, and sensors. With our new USB adapter humbly taking up only the real estate of its DB9 predecessor, attaching the sensor board to the top of the Mark III could be done without incident.

The little robot that could is ready for its next evolution, and it is sure to show up again as an able competitor with some of our future projects. SV

Recommended Websites

www.junun.org/MarkIII

http://parts.digikey.com/1/parts/1934884-mod-usb-rs232-embeded-retro-db9f-db9-usb-f.html

www.ftdichip.com

Special Thanks

Bill Woolley, for letting us commandeer his kit!

Robotics Showcase

SDP/SI
STOCK DRIVE PRODUCTS/STERLING INSTRUMENT
www.sdp-si.com
FREE CATALOG
800.819.8900
GEARS
BELT & CHAIN DRIVES
PULLEYS
COUPLINGS
BEARINGS
SPROCKETS

SMALL MECHANICAL COMPONENTS

ALL ELECTRONICS CORPORATION
THOUSANDS OF ELECTRONIC PARTS AND SUPPLIES
VISIT OUR ONLINE STORE AT www.allelectronics.com
WALL TRANSFORMERS, ALARMS, FUSES, CABLE TIES, RELAYS, OPTO ELECTRONICS, KNOBS, VIDEO ACCESSORIES, SIRENS, SOLDER ACCESSORIES, MOTORS, DIODES, HEAT SINKS, CAPACITORS, CHOKES, TOOLS, FASTENERS, TERMINAL STRIPS, CRIMP CONNECTORS, L.E.D.S., DISPLAYS, FANS, BREAD-BOARDS, RESISTORS, SOLAR CELLS, BUZZERS, BATTERIES, MAGNETS, CAMERAS, DC-DC CONVERTERS, HEADPHONES, LAMPS, PANEL METERS, SWITCHES, SPEAKERS, PELTIER DEVICES, and much more....
ORDER TOLL FREE
1-800-826-5432
Ask for our FREE 96 page catalog

Recycling & Remarketing High Technology
WEIRDSTUFF® WAREHOUSE
Software,Computers,Electronics,Equipment,Doohickies
384 W. Caribbean Dr.
Sunnyvale, CA 94089
Mon-Sat: 9:30-6:00 Sun: 11:00-5:00
(408)743-5650 Store x324
WE BUY AND SELL EXCESS & OBSOLETE INVENTORIES!
FREE COMPUTER RECYCLING
We recycle computers, monitors, and electronic equipment. M-Sat 9:30-4:00
GREAT DEALS!
Hi-tech items, electronics test equipment, and more!
GIANT AS-IS SECTION
10,000 sq. ft. of computers, electronics, software, doo-hickies, cables, and more!
also check out our...
eBay Store
stores.ebay.com/WeirdStuff-Inc
WWW.WEIRDSTUFF.COM

Then and NOW

MECHANICS FOR ROBOT HANDS AND ARMS

by Tom Carroll

Our recent series of military actions in the Middle East have resulted in many of our fighting men and women returning home with missing hands, arms, and legs. Many of these marvels of mechanics, electronics and computer science are quite complicated and literally cost an "arm and a leg," certainly no pun intended. Sometimes purely mechanical solutions seem to work best in many situations where one tends to first look at an electro-mechanical solution. It is these mechanical solutions to robot hand and arm designs that I'd like to discuss.

Imagine yourself in the field of battle. It's four years into the new century and you're 24 years old with seven years of military experience behind you. In the heat of battle, you're struck in the arm, not by a mere bullet but by a cannon round and shrapnel of broken bits of metal. You're bleeding severely and fellow soldiers help you aside and tend to you before medical help comes to your rescue. Later, to your shock, you are told that your hand has been severed and your lower arm is so mangled that it must be amputated.

You're a soldier and fighting is in your blood. What can be done to give you back your arm? One that can move? One that can manipulate things? As you slowly heal, the first prosthetic hand that you are given is nothing more than a stiff mitt with mere claws for fingers. You ask for a better hand and arm, and finally a mechanical prosthetic solution is custom-made for you with articulated fingers that can grasp and manipulate objects. Oh, the joys of modern medical science and artisans that can create mechanical miracles to assist wounded soldiers.

FIGURE 1. Gottfried Götz Von Berlichingen.

The Götz Iron Hand

Am I describing a soldier injured in one of the Middle East wars of 2004? Actually, the year is 1504 and I am speaking of Gottfried "Götz" Von Berlichingen (shown in **Figure 1**) — a soldier of fortune or mercenary for hire. **Figure 2** shows the artificial hand and lower arm that he was given. The term 'prosthetic arm' was not used back then; it was commonly called the Götz Iron Hand and was a mechanical masterpiece. Most accounts of Götz's injury speak of a cannon blast severing his hand, and shrapnel from his sword and armor severely injuring his lower arm. However, there are several accounts that speak of an altercation with a farm hand that resulted in the lower arm being cut off by a sword. Possibly Götz

FIGURE 2. The Götz iron hand.

— as a proud mercenary — preferred telling everyone that the accident occurred during an act of war rather than a barnyard fight.

Götz was born of noble background but quickly made enemies of any nobleman or person of authority with his nasty attitude. He began his career at the tender age of 17 as a soldier in the service of Margrave Frederick I in 1497 Germany. He then fought for the Emperor Maximillian I of the Holy Roman Empire for a few years after that. At the age of 20, he formed a mercenary troupe and sold his services to various royal noblemen who needed ruthless soldiers to attack their enemies. When not 'employed' by these men, he would conduct raids to plunder the wealthy, much like the mythical Robin Hood of England, but was a bit more vicious.

A recent article in *Wired Magazine* highlighted this amazing mechanical marvel of 500 years ago in an article entitled *Götz of the Iron Hand.* It made me realize just how important the need for a functional hand and arm might be to a soldier. Just this January, I discussed various types of robot hands, but these were mostly electrically-driven specifically for robots. There has been much progress in prosthetic hands and arms the past few years, and certainly in the past 500 years.

The actual hand of Götz still exists and is on display in a German museum. Götz is not only remembered to this day in Germany for the hand made for him by armor craftsmen, but for a particularly nasty phrase he often used that got him imprisoned on several occasions. This invention is often credited as the foundation of the modern prosthetics industry. The hand was fashioned with individual articulated fingers with individual springs to return them to their extended lengths. An array of levers and buttons allowed him to grasp a sword or hold a lance, hold onto the reins of his horse, write with a pen, and even play cards. He made many enemies during his long lifetime of mayhem, but managed to die peacefully in his bed in 1562 at the age of 82.

Revenge of the Nerds Robot Action Prop

Back in late 1983, I was approached by 20th Century Fox prop people about building a robot action prop (actually four) for a movie Fox was making called *Revenge of the Nerds.* They wanted a 'robot' that looked like a kit that a college freshmen nerd might build and it had to have two working arms with claws to hold things. Unfortunately, after they agreed to my designs and initial costs, they managed to wait until well into January to give me the final go-ahead. At that time, they literally threw money at me to finish the robots in several weeks, and I needed to hire several friends to build the robot props in my garage while I was at work at Rockwell. I couldn't take the time that I originally wanted to build them myself.

I used an arm configuration from an earlier promotional robot design of mine that allowed both the shoulder joint and elbow joint to move together in a natural human-like motion, without adding an extra motor. **Figure 3** shows the earlier promo configuration using a figure 8 cable arrangement and pulley system mounted on a fixed transverse tubing shaft. The top pulleys are attached to this shaft and do not rotate; only the upper arm shoulder joint rotates around the shaft.

When the upper arm/shoulder joint is moved upward by the shoulder motor — say 45º — the cable tugs at the bottom elbow pulley, also pulling it upward 45º. This resulted in a natural arm swing, much like how a human would move their arm. When the upper arm is straight down, so is the lower arm. In like fashion, when the shoulder drives the upper arm segment 90º, the lower arm segment is driven further 90º so the hand/claw faces straight up. This allows the robot to not have such a stiff robotic appearance.

Smooth arm motion is accomplished without the use of another heavy motor that would weigh down the arm. Part of the right shoulder linear actuator is barely shown in the bellows to the left, though a chain drive linked to a motor in the lower chest cavity was used in the final promo robot configuration. **Figure 4** shows the completed *Revenge of the Nerds* robot action prop using a flatter arm/pulley configuration. This robot used a direct drive to a gearmotor.

Robot Arm Counterbalancing

Another strictly mechanical feature that I've used on several large robots is the use of a coil spring (both

FIGURE 3. A 'figure 8' cable arrangement for elbow movement.

FIGURE 4. *Revenge of the Nerds* robot.

compression and extension) or a gas spring to compensate for the weight of a robot's arm mass. Many builders find that an arm of a decent length requires a fairly hefty high torque motor to even lift the arm, not including any sort of payload. For example, an arm/claw configuration that is a foot long, extended, and is weighed at the end at one pound, a servo or gearmotor of 192 (16 x 12) oz in of torque will be required to just lift the arm with no payload. That's a pretty hefty servo. Add a 16 oz payload and you're asking for 384 oz in of torque required; that's for a pretty short robot arm.

Gas Springs to the Rescue

Coil springs can work for mass compensation of robot arms but they are not linear in their force. As you compress or stretch a coil spring, more force is required the greater the distance. Gas springs (shown in **Figure 5**) are fairly linear in their force throughout their working stroke length. They are quite often used in cars to compensate for hood or trunk lid weight, as well as in applications such as photo copier and printer lids, and for medical instrument uses.

I used a set of surplus gas springs to compensate for arm weight in a large promotional robot that I made a while back. The arms were about two feet long and the weight at the end of the arm was about three pounds, or (24" x 48 oz) or 1,152 oz in of torque — again, with no payload. I placed a three inch lever inside the robot body facing down in line with the axis of the robot's arm. These levers were 1/8 the length of the arm, so I needed eight times the force on a three inch lever arm to compensate for the arm's weight, or (8 x 3 lbs) 24 pounds of force in a spring. It turned out that the gas springs that I bought had a force of about 40 pounds — more than enough to compensate for the arm's weight.

I thought a minute, "Why not over-compensate for the weight of the arm by adding a bit more upward force to give me even more payload capacity?" My shoulder gearmotor had sufficient torque to pull the arm downward with no payload attached, and the extra force allowed an even greater payload. It made the customer happy as the robot could pick up a fairly heavy six to seven pound payload. I had to make sure that the three inch levers on the arms were not allowed to poke through the shoulders, so the arm's upward motion was limited to about 15º past the horizontal, using a DPDT polarity-reversing limit switch.

Even a simple large rubber band around the shoulder shaft of a small robot's arm can be wound so that the band pulls the back surface of the shaft downward so that the arm's mass in the front will be lifted upward. I used this method in a robot that I designed for a *Boy's Life Magazine* robot construction article that I wrote in 1987. There are many mechanical ways a robot builder can jury rig a way to help a robot move its arms with a payload.

Arms that Compensate for Gravity

I was in the Portland, OR airport recently, waiting for a flight when I ran across a must-have and uniquely-titled book in a bookstore — *Packing for Mars,* by Mary Roach. She discusses many space topics that are quite interesting and few people really understand. One chapter dealt with "The Alarming Prospect of Life Without Gravity." In the early days before spaceflight, scientists and doctors were truly concerned that astronauts might suffer all sorts of maladies in space, including stoppage of the heart or loss of blood circulation. Most astronauts — after a period of adjustment — enjoy zero G, but a few have found the loss of weight on the arms a bit frustrating as they float out of control, especially after many months in space. Fortunately for these astronauts, moving objects of a large mass are fairly easy to accomplish, though acceleration and deceleration of that mass is

FIGURE 5. Gas springs from Ace Controls.

the same in space as on Earth. The astronauts themselves do need to be firmly attached to a non-movable wall, floor, or boom so they will not be the objects that move.

Terrestrial Approach to Gravity Compensation

Most people think of the Iron Man suits from the movie of the same name or the exoskeleton that Sigourney Weaver fought with in *Alien* as the best representatives of mechanisms people wear over their bodies to augment their physical capabilities. Most exoskeletons require an external energy source from a trailing cable to power the many hefty motors within the suit. Complex actuators and feedback sensors are fed or feed data to a computer, and hand controls determine the motions required. These are complex and expensive robotic machines and are designed to give the user power beyond what he might have as a mere human being. **Figure 6** shows actor Robert Downey, Jr. on the set of the Marvel Comics 2008 *Iron Man* movie discussing props used in a scene. The complexity shown in the movie is not far off from what real systems would require.

Compensating for gravity does *not* mean adding superior strength to a human's lifting capacity. Workers in automobile assembly lines quite often use overhead cables to counterbalance a particularly heavy tool that needs to be maneuvered around a car during assembly, such as a spot-welding head or a frame to pick up and install a windshield or dashboard. The parts do not need to be heavy to tire a person from repetitive handling. Sometimes a human just needs a bit of help to *continue* lifting or holding an object, such as in smaller assembly-line operations. A worker might need a bit of help retrieving parts out of bins right in front of him (or her) hundreds of times a day or more.

Mechanical Solution to Overcome Gravity

Many times, movie studios use steel track laid down on the ground over which a camera dolly is pushed to follow a moving scene. A handheld camera can do the same job without the task of laying track. Handheld cameras have been used for years to allow a cameraperson to follow a scene while walking. This is especially true when following the action might catch a view of the track on the ground. The spring-balanced Steadicam has solved this dilemma. A fairly affordable Steadicam "Glidecam Smooth Shooter" is shown in **Figure 7** in use with a video camera.

FIGURE 6. Robert Downey, Jr. discusses Iron Man props on set.

A company called Equipois has utilized this technique for other applications, especially small parts assembly operations. Their 'Defy Gravity' devices shown in **Figure 8** can be attached to a person's chair, a wall, or any nearby location to assist a person in any type of repetitive lifting task. The **figure** shows two examples: one which appears to be holding an air-driven tool, and the other an open ring that can be adapted to hold a round object.

The device does not come ready to use for every potential application. It must be adjusted for the user's arm weight, strength, potential reach, and payload weight. The series of springs can be tightened or loosened, according to the task at hand. The Defy Gravity arms show the series of two parallelograms, both fitted with a spring that twists each parallelogram in the opposite direction in proportion to the gravity (and payload) trying to distort it.

The x-Ar by Equipois shown in **Figure 9** is one of their latest products that can be worn as a back brace-like arrangement, mounted on a work stand, wall track, or

FIGURE 7. The Steadicam Glidecam Smooth Shooter.

FIGURE 8. Defy Gravity by Equipois.

FIGURE 9. x-Ar by Equipois.

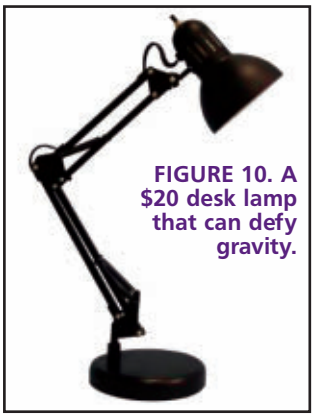


FIGURE 10. A $20 desk lamp that can defy gravity.

other apparatus beside the worker. The x-Ar is not powered like a complex exoskeleton, but uses simple springs to compensate for expected loads. The use of this simple device makes even more sense to use if only a single arm is required. The x-Ar is positioned on the dominate arm of the user — the one he/she would normally use in the repetitive assembly operation.

In our homes, we may have a desk lamp over a workbench or drawing table like the one shown in **Figure 10** that has a series of parallel steel rods at several hinge points and some compensating springs. At less than $20, this gravity-compensation technology is not too expensive to use even on the tightest robot construction project where a robot's arm needs a bit of help overcoming gravity.

Final Thoughts

Arms add a special human touch to robots; even more so with hands. Using a mechanical technique to improve your robot's ability to lift a payload, is not only cheaper than the use of a motor, shaft encoder, or necessary microcontroller control software, but draws no extra power from a limited power source. Go through your junk boxes and pull out those coil springs and gas springs, rods, hinges, or whatever, and just sit down and piece together a masterpiece for mechanical buddy.

I encourage all of you who may want to explore the addition of arms and hands to your creations to look at simple mechanical means to assist these appendages. If a wild mercenary of 500 years ago could do it, imagine what you can do with today's mechanical technologies. SV

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links.

Advertiser Index

I2I CONTROLS

Tomorrow's programming revolution, starts today!

The choice is yours to make

```
//-----------------------------------------
*
SSD1963_8-BIT
PROGRAM FOR WRITING TO NE-
WHAVEN DISPLAY 5.7" TFT
*/
//-----------------------------------------
DELAY_MS(100);
WRITE_COMMAND(0X01); //SOFTWARE RESET
WRITE_COMMAND(0X01);
WRITE_COMMAND(0X01);
DELAY_MS(10);
COMMAND_WRITE(0XE0,0X01); //START PLL
COMMAND_WRITE(0XE0,0X03); //LOCK PLL
WRITE_COMMAND(0XB0); //SET LCD MODE SET TFT 18BITS
MODE
WRITE_DATA(0X0C); //SET TFT MODE & HSYNC+VSYNC+DEN
MODE
WRITE_DATA(0X80); //SET TFT MODE & HSYNC+VSYNC+DEN
MODE
WRITE_DATA(0X01); //SET HORIZONTAL SIZE=320-1 HIGHT-
BYTE
WRITE_DATA(0X3F); //SET HORIZONTAL SIZE=320-1 LOWBYTE
WRITE_DATA(0X00); //SET VERTICAL SIZE=240-1 HIGHTBYTE
WRITE_DATA(0XEF); //SET VERTICAL SIZE=240-1 LOWBYTE
WRITE_DATA(0X00); //SET EVEN/ODD LINE RGB SEQ.=RGB
COMMAND_WRITE(0XF0,0X00); //SET PIXEL DATA I/F
FORMAT=8BIT
COMMAND_WRITE(0X3A,0X60); // SET R G B FORMAT = 6 6 6
WRITE_COMMAND(0XE6); //SET PCLK FREQ=4.94MHZ ; PIXEL
CLOCK FREQUENCY
WRITE_DATA(0X02);
WRITE_DATA(0XFF);
WRITE_DATA(0XFF);
WRITE_COMMAND(0XB4); //SET HBP,
WRITE_DATA(0X01); //SET HSYNC TOTAL 440
WRITE_DATA(0XB8);
WRITE_DATA(0X00); //SET HBP 68
WRITE_DATA(0X44);
WRITE_DATA(0X0F); //SET VBP 16=15+1
WRITE_DATA(0X00); //SET HSYNC PULSE START POSITION
WRITE_DATA(0X00);
WRITE_DATA(0X00); //SET HSYNC PULSE SUBPIXEL START POSI-
TION
WRITE_COMMAND(0XB6); //SET VBP,
        MORE
```

```
; USING NEWHAVEN DISPLAY
WITH SSD1963 CONTROLLER

; DO DATE AND TIME

SETIO G, 0XF000 ;SET LOCATION FOR DISPLAY
SETIO U, 0XFA00 ;SET LOCATION FOR USB DRIVE
LOAD "FONT.HEX" ;GET FONT TABLE FROM USB
SETGLCD I  ;INITIALIZE THE GRAPHICS LCD
SETGLCD C  ;CLEAR THE DISPLAY
SETGLCD D1 ;SELECT LARGE FONT

DO

 SETGLCD P,(100,200);SET CURSOR POSTION

; DISPLAY TIME
 GLCDOUT USING(2),!H,":",!M,":",!S,"  ",

; DISPLAY DATE
 GLCDOUT USING 2),!MO,"/",!D,"/",!Y

UNTIL S=1


; ALL DONE
```

Choose your path at

www.i2icontrols.com

SERVOCITY

A DIVISION OF ROBOTZONE, LLC.

WE HAVE THE PARTS FOR YOUR IDEAS!

our pan & tilt kits are perfect for your unique project!

See videos of all the products shown below in action on YouTube.

only 0.29 oz!

- 48 pitch gears
- 2 : 1 ratio
- Pan rotates 180°
- Tilt moves 90°
- Quick assembly

$19.99
SPT50 Micro Pan & Tilt Kit
Servos not included

- Payload up to 10 oz.
- 135° tray swing
- Can be mounted hanging, upright or sideways!

Great for R/C aircraft applications!

$19.99
SPT100 Pan & Tilt Kit
Servos not included

GoPro
This system works GREAT with GoPro cameras!

- Payload up to 2 lbs!
- Easy mounting

Incorporates 1/4" shafts supported by dual ball bearings and 1/4" ABS plastic to help make it the most rigid direct drive pan and tilt available.

$45.99
SPT200 Pan & Tilt Kit
Servos not included

- 48 pitch gears
- Choose your gear ratio (up to 8.6 : 1)
- 135° of tilt
- No slippage

Patented

Shown with 8.6 : 1 ratio, full metal gears

AS LOW AS **$72.94** INCLUDING SERVO
SPT400 Tilt System

- Available with 4" or 6" cradle
- Payload up to 4 pounds!
- Mount hanging, upright or sideways

DDT540 4"

DDT560 6"

STARTING AT **$39.99**
DDT 540 / 560 Tilt Kits
Servos not included

rotate over 75 lbs!

- Up to 300° rotation
- No slippage
- Comes fully assembled
- 915 oz-in of power!
- 6.90 sec / 360°
(5:1 ratio)

Patented

AS LOW AS **$159.99** FULLY ASSEMBLED W/ SERVO
Bottom Mount Pan Systems

FULL LINE OF HITEC SERVOS • SERVO CONTROLLERS • BATTERIES & POWER SUPPLIES • SERVO EXTENSIONS • SERVO POWER GEARBOXES • HEAVY DUTY P&T SYSTEMS

Like us on Facebook

To view our entire selection of products visit us online at

WWW.SERVOCITY.COM

Specializing in Servos, Radios, Motors, Pan & Tilt Systems, Batteries, Wiring, Connectors, Gearboxes and more!

Check out our videos on YouTube
www.youtube.com/user/ServoCity

Copyright 1999-2011 Robotzone, LLC. - All Rights Reserved
ServoCity is a registered trademark of Robotzone, LLC

Phone: (620) 221-0123
E-mail: Sales@ServoCity.com

Prices and availability subject to change without notice.

SUNSTONE CIRCUITS

THE EASIEST PCB COMPANY TO DO BUSINESS WITH

ValueProto™ PCBexpress® Full Feature

Sunstone Circuits® pioneered the online ordering of printed circuit boards and is the leading PCB solutions provider with more than 35 years of experience in delivering quality prototypes and engineering software. With this knowledge and experience, Sunstone is dedicated to improving the PCB prototyping process from quote to delivery (Q2D®).

Did You Know? Sunstone Offers:

- Controlled impedance testing
- Fine lines and spacing [.003]
- RoHS compliant finishes
- Free 25-point design review
- Free shipping & no NRE's
- Flex / Rigid Flex Boards
- Online Quote & Order
- PCB123® design software
- RF / Exotic Materials
- Over 99% on-time or early delivery
- Best PCBs in the industry
- Live customer support 24/7/365